UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

REUTILIZAÇÃO BASEADA EM CASOS DE EXPERIÊNCIA NA ÁREA DE MENSURAÇÃO EM ENGENHARIA DE SOFTWARE

Tese submetida à Universidade Federal de Santa Catarina para a obtenção do título de Doutor em Engenharia de Produção.

Christiane Gresse von Wangenheim

Florianópolis, Brasil

Abril 2000

Christiane Gresse von Wangenheim

Reutilização Baseada em Casos de Experiência na Área de Mensuração em Engenharia de Software

Esta tese foi julgada adequada para a obtenção do título de Doutor em Engenharia de Produção e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina.

Prof. Ricardo Miranda Barcia, Ph.D.
Coordenador do Curso

Banca Examinadora:

Prof. Ricardo Miranda Barcia, Ph.D.
Orientador

Prof. Jose Augusto Mariz de Mendonça, Dr.
Examinador Externo

Prof. Marcelo Thiry Comicholi da Costa, Dr.
Examinador Externo

Prof. Alejandro Martins, Dr.
Membro

Prof . João Bosco da Mota Alves, Dr.
Membro/Moderador

Florianópolis, 6 de Abril de 2000

Agradecimentos

Ao professor Ricardo M. Barcia pela orientação, recursos e tempo dedicado para poder desenvolver e concluir esta pesquisa.

Aos Professores Membros da Banca, especialmete aos professores externos, Prof. Mariz de Mendonça e Prof. Marcello Thiry, pela dedicação e os comentários e sugestões construtivos ajudando no melhoramento deste trabalho.

Aos professores, colegas e funcionários do Departamento de Engenharia de Produção e Sistemas, Departamento de Informática e Estatística e do Centro GeNESS da UFSC por todo o apoio e discussões frutíferas durante estes anos de estudo. Especialmente ao Prof. Alejandro Martins pelo suporte comprehensivo e permanente e ao Professor Roberto Pacheco, cujas críticas e idéias foram muito valiosas. Agradeço também Prof. João Bosco por toda a confianca e motivação dadas ao meu trabalho.

Pelo apoio e dedicação de Prof. H. Dieter Rombach, Klaus-Dieter Althoff, Prof. Lionel Briand e Guenther Ruhe, do Fraunhofer Institute for Experiemental Software Engineering de Kaiserslautern, que sempre souberam conduzir meus esforços, e também aos meus colegas do Software-Technology-Transfer Initiative e do Fraunhofer Insitute for Experimental Software Engineering pela cooperção efetiva.

Também não poderia deixar de agradecer aos professores Vic Basili e Michael M. Richter pelas discussões e comentarios valiosos ao meu trabalho.

Desejo ainda agradecer aos estudantes do mestrado Marcos R. Rodrigues, Marcel Pacheco, Lurregi Correa e André Bortolon pelas idéias e pela dedicação durante todo o correr este trabalho.

Aos estudantes da disciplina de Mensuração de Software do 3. trimestre 1999 do Programa de Pós-Gradução em Engenharia de Produção da UFSC pela participação no estudo empírico e a Prof. Walter Cybis do Departamento de Informática da UFSC por seu apoio durante o planejamento do experimento.

Gostaria de agradecer também às empresas nas quais participei da aplicacão da tecnologia de mensuração, que estimularam e compartilharam suas experiências, o que contribuiu

enormemente para os resultados desta pesquisa.

À Aline R. Alves pela tradução do meu "portingles", que me ajudou muito na redação final da Tese.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES, pela indispensável ajuda financeira.

Ao meu marido Aldo, que sempre me motivou nos momentos de incerteza, e ajudou e cooperou tanto com a minha pesquisa. Obrigada por seu amor, carinho e sua compreensão, fazendo a vida um paraíso.

E, por fim, a todas as pessoas que participaram, direta, ou indiretamente, na conclusão deste trabalho.

Resumo

Inúmeras tecnologias em Engenharia de Software têm sido desenvolvidas durante os últimos anos para a melhoria da qualidade e produtividade de software. Essas tecnologias, em geral, fornecem uma representação conceitual explícita das tarefas a serem desenvolvidas. Essa representação é conveniente para a sumarização e comunicação do conhecimento de tarefas complexas. Entretanto, quando há a transferência de tecnologias inovadoras em Engenharia de Software na indústria, estas devem ser adaptadas às características específicas e às necessidades da organização em particular e desenvolvidas de acordo com uma maneira evolutiva e crescente, baseada no contínuo resultado obtido da sua aplicação na prática. Desse modo, as representações de tecnologias têm grande probabilidade de serem simplificadas, e freqüentemente não incluem aspectos referentes à sua aplicação e conhecimentos de como utilizá-la na prática. Neste contexto, o conhecimento tácito de como aplicar e adaptar tecnologias de Engenharia de Software na prática em forma de conhecimento baseado em experiências torna-se uma importante fonte de aprendizado organizacional. Reutilizando conhecimento baseado em experiência representando lições aprendidas pela descrição de problemas ocorridos em aplicações passadas dessas tecnologias e soluções aplicadas pode levar à resposta frente a novas situações baseada em experiências similares realizadas em aplicações anteriores. A reutilização do conhecimento baseado em experiências pode prevenir a repetição de falhas e guiar a solução de problemas, resultando em economia de tempo e de custos. Além disso, a criação de competências de software específicas à organização promove a ampla difusão e uso efetivo de novas tecnologias na prática. O *feedback* contínuo obtido da aplicação dessas tecnologias ajuda na melhoria sistemática e na adaptação de tecnologias de Engenharia de Software para melhor satisfazer necessidades práticas. Portanto, conhecimento baseado em experiências que descreve lições aprendidas realizadas em um projeto de software específico tem que ser sistematicamente coletado e disponibilizado na organização para futura reutilização. O objetivo deste trabalho de pesquisa é desenvolver, implementar e avaliar uma metodologia utilizando *Raciocínio Baseado em Casos*. Esta metodologia visa a representação, armazenamento, recuperação e contínua aquisição e integração de lições aprendidas em uma Base de Casos. O enfoque é aplicado no contexto do paradigma *Goal/Question/Metric*, uma nova abordagem sobre mensuração de software, a qual auxilia na definição e implementação de metas operacionais e mensuráveis para melhoria de software. A extensão desta abordagem através da possibilidade de reutilizar sistematicamente lições aprendidas mostrou neste trabalho contribuir significativamente para o aperfeiçoamento do processo criativo de planejar programas de mensuração e levar a uma redução substancial de esforços.

Abstract

Numerous Software Engineering technologies have been developed during the last years for the improvement of software quality and productivity. These technologies, in general, provide an explicit conceptual representation of the tasks to be performed. This representation is convenient for summarizing and communicating complex task knowledge. However, while transferring innovative Software Engineering technologies into industry, they have to be tailored to the specific characteristics and needs of a particular organization and developed in an incremental and evolutionary manner based on continuous feedback from their application in practice. Therefore, the technology representations are likely to be simplified, often do not include aspects of their application circumstances and knowledge on how to use these technologies in practice. In this context, tacit knowledge on how to apply and adapt Software Engineering technologies in practice in form of experiential knowledge becomes an important source for organizational learning. Reusing experiential knowledge representing lessons learned by describing problems which occurred in past applications of these technologies and applied solution strategies can guide responding new situations based on similar experiences made in previous applications. Reusing experiential knowledge can prevent the repetition of past failures and guide the solution of actually occurred problems, resulting in cost and time savings. Furthermore, the creation of organization-specific software competencies promotes the widespread effective use of innovative technologies in practice. Continuous feedback from their application helps the systematic enhancement and tailoring of software engineering technologies to better meet practical needs. Therefore, experiential knowledge describing concrete lessons learned made in individual software projects has to be systematically gathered and made available organization-wide for future reuse. The objective of this research work is to develop, implement and evaluate a methodological framework based on *Case-based Reasoning* methods for the representation, storage, retrieval and the continuous acquisition and integration of lessons learned in a case base. The approach is applied in the context of the *Goal/Question/Metric* paradigm, an innovative software measurement approach, which helps defining and implementing operational and measurable software improvement goals. Enhancing the approach by the possibility to systematically reuse measurement lessons learned was shown in this work to significantly contribute to the improvement of the creative process of planning measurement programs and lead to substantial effort reductions.

Índice

Lista de Siglas

GQM	Goal/Question/Metric
GQM-LL-EF	GQM-Lessons Learned-Experience Factory (Fábrica de Experiência de Lições Aprendidas de GQM)
GQM-LL-KB	GQM-Lessons Learned-Knowledge Base (Base de Conhecimento de Lições Aprendidas de GQM)
GQM-PEC	GQM-Product Experience Case (Caso de Experiência de Produto de GQM)
GQM-PSEC	GQM-Problem/Solution Case (Caso de Problema/Solução de GQM)
GQM-R+	Processo de planejamento GQM com reutilização
E-MOP	Memória Episódica
ES	Engenharia de Software
FE	Fábrica de Experiencia (Experience Factory)
GE	Episódios Generalizados
PPGEP	Programa de Pós-Graduação em Engenharia de Produção
QIP	Quality Improvement Paradigm
RBC	Raciocínio Baseado em Casos
RI	Recuperação de Informação
SGBD	Sistemas de Gerenciamento de Banco de Dados
SW	Software
UFSC	Universidade Federal de Santa Catarina

1 Introdução

Hoje, quase todas as organizações estão envolvidas no desenvolvimento ou uso de software. A qualidade do software tem uma importância essencial para a competitividade de uma organização no mercado. Mas, na realidade, o processo de desenvolvimento e manutenção de software é freqüentemente insuficiente com respeito à qualidade, produtividade e preditibilidade [Jal97,Gib94].

O enfoque primário de uma organização é desenvolver um produto de alta qualidade dentro de um cronograma e um orçamento. Por causa dos requisitos permanentemente crescentes no desenvolvimento de software com respeito ao cliente ou novas tecnologias, o melhoramento sistemático e contínuo precisa ser parte integrada do processo de software. O melhoramento do produto é tipicamente atingido através do melhoramento dos processos usados para produzir o produto. O melhoramento dos processos pode ser atingido pela modificação dos processos gerênciais ou técnicos, ou pela introdução das novas tecnologias. Fundamental para o melhoramento sistemático é a compreensão e o estabelecimento dos níveis correntes com relação à situação atual na empresa [NAS94]. Isso inclui p.ex. entender os seguintes aspectos:

- Como é distribuído o esforço no desenvolvimento e manutenção de software na empresa?
- Quanto esforço é gasto em correção de defeitos?
- Qual é a percentagem dos defeitos detectados em inspeções?

O primeiro passo para enfocar esses aspectos é estabelecer uma linha-mestra na organização com respeito aos produtos e processos de software [DR96,BC95]. Baseada na informação quantitativa ganha por um programa de mensuração, uma organização pode construir seus modelos específicos e examinar fatores de contexto que poderiam ter um impacto sobre esses modelos. Por exemplo, o esforço do desenvolvimento de software pode depender do tamanho do sistema desenvolvido e da experiência da equipe. A compreensão completa da combinação complexa dos vários fatores de influência só pode ser atingida por uma avaliação dentro de uma organização através de programas de mensuração específicos.

Essa compreensão dos processos e práticas na organização é essencial para planejar,

controlar e melhorar o desenvolvimento e manutenção de software. Um objetivo dos modelos organizacionais com respeito aos produtos, processos de software e seus fatores de influência é prover melhor informação (quantitativa) para decisões gerenciais. O conhecimento ganho através da análise e interpretação dos dados de mensuração pode ser usado para [NAS94]:

- Planejamento de novos projetos de software com base na estimativa de esforço requerido, cronograma, alocação de equipe, etc. Através dos modelos formais derivados nos programas de mensuração passados, conhecimento sobre esses aspectos de projetos semelhantes passados podem ser reutilizados para um planejamento mais informado e apropriado.
- Controle do projeto corrente através da comparação dos valores estimativos com os valores atuais coletados pelo programa de mensuração acompanhando o projeto corrente. Resultando da monitoria do estado do projeto, ajustes e ações corretivas podem ser iniciadas, quando desvios do plano de projeto ou problemas aparecem.
- Validação dos modelos organizacionais desenvolvidos em programas de mensuração passados. Por causa das mudanças permanentes na área de software, p.ex. com respeito às novas tecnologias, os modelos precisam ser ajustados para novas tendências ou mudanças no ambiente.

Enfocando o melhoramento da qualidade, os pontos fortes e fracos podem ser identificados pelo programa de mensuração inicial. Com base nessa compreensão inicial, áreas de melhoramento potenciais com alto impacto na qualidade do produto podem ser identificadas e ações de melhoramento podem ser selecionadas, executadas e avaliadas. O impacto dessas mudanças precisa ser assessorado quantitativamente pela mensuração acompanhando as modificações. Ele só pode ser determinado, se uma linha-mestra quantitativa está disponível, contra a qual as comparações podem ser feitas. De outra maneira é impossível determinar se uma mudança obteve um efeito positivo ou não [DHL95].

Como quantificar?

Muitas organizações estabeleceram processos de desenvolvimento e manutenção de uma maneira *ad-hoc*, baseadas em experiências de outras empresas. Isso resultou muitas vezes em soluções incompatíveis, conseqüências negativas e equipes sem motivação [Rom91]. Para

melhorar a qualidade e produtividade de software na indústria, é necessário entender o software e os processos de desenvolvimento e manutenção de software específicos de uma empresa. O problema com o software, é que seu desenvolvimento é influenciado por um espectro extremamente grande de fatores como: de processo, produto ou recursos. Para adquirir uma visão mais profunda deste processo de desenvolvimento e dos fatores de influência, nós necessitamos de modelos formais e de uma forma de organização para o reuso de todo o conhecimento [BCR94a]. É especialmente importante definir o nível corrente de compreensão de processos e produtos[1] de software em uma empresa de uma forma quantitativa. De forma que as metas do negocio da empresa possam ser julgadas se estão satisfeitas ou não. Isso só pode ser obtido em uma empresa através da modelagem e da mensuração explícita dos fatores que influenciam o processo de produção de software [Rom91].

O que é mensuração?

Mensuração visa a captura de características de entidades e seus interrelacionamentos, bem como a sua manipulação de modo formal [KPF95]. Medidas de software são usadas para quantificar certos atributos do processo, produto e recursos de software. Por exemplo, pode-se utilizar medidas para analisar a efetividade de tecnologias, produtividade e qualidade de produtos. (por exemplo, confiabilidade, usabilidade, manutenibilidade, etc).

A mensuração envolve o mundo real empiricamene no qual esta ocorre e um modelo formal baseado em aspectos do mundo real associado às medidas apropriadas [Zus98,FP97,OJ97,BEM96]. Na mensuração, valores são associados aos atributos de entidades no mundo real com o objetivo de adquirir uma meta pre-definida representada no modelo empírico. O modelo formal fornece uma base abstrata para a coleta e análise de dados de mensuração. Para a mensuração de software efetiva e uma interpretação significativa de dados resultantes, aquelas dependências entre o modelo empírico e o modelo formal devem ser consideradas, preservando o entendimento a respeito de entidades e relacionamentos

1. Com processo de software nós denotamos os processos de desenvolvimento e manutenção de software e cada sub-processo deles, p.ex., análise de requisitos ou fase de testes. Com produto de software nós denotamos todos os produtos desenvolvidos durante o processo de software, p.ex. código, documentação dos testes [Jal97].

básicos que existem entre as entidades e atributos no mundo real.

Metodologias

Existe uma grande variedade de dados que podem ser coletados (tais como dados de recursos, mudança e defeito, processo e produto) e muitas maneiras diferentes pelas quais a mensuração pode ser feita. (Por exemplo, o tamanho de um módulo de software pode ser medido através do número de linhas de código, cálculo de pontos de função, ou através da soma do número de métodos e objetos) [Bas99].

Nesse contexto, o paradigma *Goal/Question/Metric Paradigm* (GQM) [BDR97,BCR94b,BW84] é uma abordagem orientada a metas para a mensuração de produtos e processos em Engenharia de Software, suportando o definição *top-down* de um programa de mensuração e a análise e interpretação *bottom-up* dos dados de mensuração. Ele foi utilizado com sucesso em diversos ambientes, como NASA-SEL (EUA) [BCM+92], Robert Bosch GmbH (Alemanha) [BDT96], Allianz Lebensversicherungs-AG (Alemanha) [GRR94], Digital SPA (Itália) [FLM96], Motorola [Das92] e Schlumberger (Holanda) [LSO+98,HOL+96,SLO95]. O paradigma GQM é baseado no requisito de que a mensuração deveria ser orientada a metas, p.ex., toda coleta dos dados deve ser baseada num fundamento lógico, que é documentado explicitamente. Essa abordagem tem várias vantagens: ela suporta a identificação das medidas úteis e relevantes tanto como suporta a análise e interpretação dos dados coletados. Ela permite um assessoramento da validade das conclusões a que se chegou e evita a resistência contra programas de mensuração pela equipe do projeto de software.

Para apresentar essas vantagens, programas de mensuração baseados em GQM devem ser planejados e executados de acordo com os seguintes princípios [Rom91]:

- A tarefa de análise a ser executada precisa ser especificada precisamente e explicitamente através de uma meta de mensuração.
- Medidas precisam ser derivadas de uma forma *top-down* baseada em metas e perguntas. Uma estrutura de metas e perguntas não pode ser adaptada de forma retroativa a um conjunto de medidas existente.
- Cada medida precisa ter um fundamento lógico subjacente que é documentado

explicitamente. O fundamento lógico é usado para justificar a coleta dos dados e para guiar a análise e interpretação.

- Os dados que são coletados com respeito às medidas precisam ser interpretados de uma forma *bottom-up* no contexto das metas GQM e das perguntas. Isso dá suporte à interpretação dos dados para as limitações e suposições do objeto através de um fundamento lógico de cada medida.
- As pessoas que vão utilizar os resultados do programa de mensuração e a partir de cujo ponto de vista a meta de mensuração é formulada, precisam ser envolvidas profundamente na definição e interpretação do programa de mensuração. Elas são os reais peritos com respeito ao objeto e ao enfoque de qualidade investigado no programa de mensuração e portanto provêem interpretações válidas no ambiente especifico.

Como montar um programa de mensuração?

Para serem efetivos e eficientes, programas de mensuração têm que ser adaptados às características de uma organização específica, aos processos e aos produtos de software, juntamente com os objetivos de negócio da empresa. Portanto, um dos maiores fatores de sucesso na mensuração é seu planejamento apropriado. Uma vez que é uma tarefa intelectualmente complexa e que requer consumo de recursos e pessoas com experiência, um maior suporte para o estabelecimento da mensuração baseada em GQM na prática faz-se necessário. Entretanto, já que a Engenharia de Software é uma disciplina nova, suas tecnologias não são tão maduras quanto outras disciplinas de engenharia. O problema de se transferir o enfoques inovadores de Engenharia de Software na prática industrial é a falta de experiência de como aplicar, utilizar e adaptar as tecnologias em diferentes circunstâncias [PJC+97].

Assim, para a contínua melhoria e difusão da aplicação de tecnologias de Engenharia de Software inovadoras tais como, enfoques de mensuração de software, captura e reutilização explícita de experiências em tecnologias de engenharia de software adaptadas à características específicas e necessidades de uma organização em particular é essencial [BCR94a] e mostra exibir um considerável potencial de crescimento [ZS95,PH90].

Os benefícios da reutilização de software são muitos. Entre eles estão a melhoria da produtividade, confiabilidade, melhores estimativas e mais rápida distribuição do produto ao mercado [SPD+94]. Tradicionalmente, na área de Engenharia de Software, a ênfase está na reutilização de código. Entretanto, com o objetivo de melhorar continuamente a qualidade e produtividade, e utilizar novas tecnologias de Engenharia de Software na prática, muitos tipos de conhecimentos organizacionais relacionados a software podem ser reutilizados. [GB97a, GRA+98, Hen97].

Segundo Polanyi [Pol66] e Nonaka e Takeuchi [NT95], o conhecimento organizacional pode ser classificado em conhecimento tácito e conhecimento explícito. O conhecimento tácito é o conhecimento pessoal que reside na experiência de um indivíduo, envolvendo fatores intangíveis como crença pessoal, perspectiva e valores e é específico ao contexto. O conhecimento explícito refere-se ao conhecimento transmissível em linguagem formal e sistemática. Os dois tipos de conhecimento são mutualmente complementares e interagem um com o outro. Com o objetivo de suportar a aplicação de tecnologias de engenharia de software, especialmente o conhecimento tácito é de interesse, pois representa o conhecimento prático acerca de como realizar coisas. No entanto, o conhecimento tácito é difícil de ser formulado e comunicado e é muito mais intercambiado através de contato direto [NT95]. Ele deve ser destacado, o que significa que ele precisa ser articulado e tornado explícito de maneira a se tornar um recurso organizacional. Conseqüentemente, o conhecimento tácito foi negligenciado na maioria das vezes no passado, embora a interação entre essas duas formas de conhecimento é a dinâmica chave da criação do conhecimento [NT95].

Estudos recentes [CM96] propõem que o conhecimento experimental [Nor93] na forma de memórias passadas, é uma importante fonte de conhecimento que contribui para o aprendizado. O conhecimento experimental descreve experiências concretas obtidas por indivíduos, por exemplo, pela observação de tentativas e erros. Conhecimento experimental pode servir como guia para reagir frente a novas situações reutilizando soluções que foram utilizadas no passado para resolver situações semelhantes, ao invés da elaboração de uma nova solução a cada nova situação. A consideração do conhecimento experimental também oferece uma possibilidade de se converter o conhecimento tácito em conhecimento explícito, que pode ser mais facilmente compartilhado pela organização. Assim, realçar tecnologias pelo

conhecimento experimental, descrever como as tarefas devem ser executadas enquanto há a quantificação de objetivos específicos e características de um projeto de software ou organização, podem facilitar substancialmente sua aplicação na prática [Gre98]. A reutilização do conhecimento experimental pode prevenir a repetição de falhas do passado e servir como guia para a solução de problemas atuais. A diminuição do número de problemas e utilização de soluções mais eficientes resultam em economia de tempo e de custos. Além disso, o contínuo *feedback* por parte da aplicação das tecnologias de ES possibilitam sua melhoria e adaptação evolutivas, criando capacidades de software específicas à organização, as quais promovem o efetivo e disseminado uso de tecnologias inovadoras na prática.

Reutilizando experiências

Capturar e reutilizar este conhecimento experimental pode melhorar significativamente a qualidade e produtividade de software. Entretanto, o que representa relevante *know-how* em software, difere de uma companhia para outra, dependendo do contexto e de seus objetivos específicos. Assim, *know-how* em software específico de uma organização, o qual compreende o centro de uma organização madura, deve evoluir continuamente com base em experiências obtidas em projetos de software. Nesse sentido, empresas de software necessitam suporte para continuamente coletar experiências de seus projetos, armazenando tais experiências, validando e reutilizando tais experiências em projetos futuros.

Atualmente, a reutilização de conhecimentos de Engenharia de Software é feita de maneira ad-hoc, informal, geralmente limitada a experiências pessoais. Para uma aquisição sistemática e comunicação dessas experiências na empresa, memórias corporativas devem ser construídas[ABT98,Gre98,Hen97,BCR94a].

Para a contínua melhoria da qualidade e produtividade de software, a captura e utilização de experiências em tecnologias de Engenharia de Software é essencial. Esse conhecimento experimental pode ser representado em forma de lições aprendidas declarando explicitamente um problema que ocorreu e a estratégia adotada para resolvê-lo.

Com o objetivo de organizar o aprendizado por experiências em ambientes industriais, memórias corporativas para uma aquisição sistemática e uma ampla comunicação do

conhecimento experimental na organização têm que ser coletadas [BM96,KS96,BCR94a]. Como uma estrutura lógica e física destinada ao contínuo acumulo de *know-how* em software, a abordagem da Fábrica de Experiência (FE) [BCR94a,BR88] tem sido considerada uma solução de sucesso. O enfoque FE (ver Figura 1) apresenta uma infra-estrutura para análise e síntese para todos os tipos de experiências, atuando como um repositório e suprindo tais experiências em projetos sob demanda. A Fábrica de Experiência em uma empresa

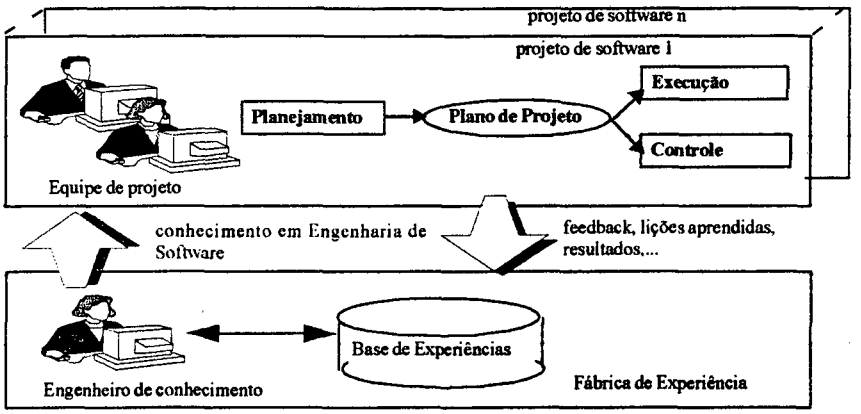


Figura 1 Organização da fábrica de experiências

complementa a organização do projeto ao permitir o contínuo aprendizado em desenvolvimento de software através de exemplos obtidos em projetos de software individuais e a pela comunicação do conhecimento de software dentro da empresa. Essas experiências na empresa podem ser de todos os tipos de conhecimento que podem apoiar o planejamento e a execução de projetos de software. Isto inclui, por exemplo, perícias e lições aprendidas sobre as tecnologias de Engenharia de Software (por exemplo, em forma de orientação ou sob forma de certo e errado), modelos de qualidade (ex. modelo de falhas), modelos de recursos (ex. distribuição de esforços), resultados e produtos (ex. documentos necessários, código, documentação) ou modelos de processo (ex. em forma de guias de processos ou material de treinamento).

Como operacionalizar a reutilização?

Como o objetivo de operacionalizar a FE no domínio da mensuração de software,

métodos sofisticados têm que ser desenvolvidos. Esses métodos devem permitir a representação e armazenamento do conhecimento baseado em experiência de modo que possa ser reutilizado, a recuperação e reutilização de experiências em mensuração em projetos futuros, e a aquisição e integração de experiências recém coletadas em uma base de conhecimento. Para alcançar este objetivo, um sistema de aprendizado baseado em conhecimento deve ser desenvolvido como uma plataforma de suporte integrada, operacionalizando a abordagem da Fábrica de Experiência na indústria. Nesse contexto, *Raciocínio Baseado em Casos* (RBC) [Wes95,AP94,Kol93], foi recentemente considerado como uma tecnologia chave para a operacionalização de uma FE [GT99,ABG+98, Gre98,Hen97,BM96], pois fornece um amplo suporte para o desenvolvimento de sistemas de aprendizado baseados em conhecimento. Raciocínio Baseado em Casos é inspirado no modelo de cognição humana, apresentando o conhecimento em forma de exemplos concretos. Ao contrário de depender inteiramente do domínio geral de conhecimento, ou fazer associações em relacionamentos generalizados, como em outros enfoques importantes de Inteligência Artificial, RBC utiliza inicialmente o conhecimento concreto adquirido em situações anteriores. Um novo problema é resolvido através da procura de experiências passadas semelhantes e sua reutilização na nova situação. A maior vantagem do RBC no contexto da FE é a recuperação baseada em similaridade para todos os tipos de conhecimento e seu aprendizado progressivo como parte integrada do processo de reutilização. Além disso, RBC pode ser estendido utilizando-se representações de conhecimento adicionais e capacidades de aprendizado [Alt96].

Entretanto, para a operacionalização da FE no domínio da mensuração de software, o RBC ainda tem que ser adaptado a necessidades específicas com o objetivo de fornecer um suporte efetivo e eficiente para o planejamento de programas de mensuração de software. Isto não é uma tarefa trivial, já que relevante conhecimento experimental na abordagem GQM tem que ser identificado, modelado e representado na Base de Experiência. Métodos para recuperação baseada em similaridade fornecendo suporte para diferentes processos, objetivos e ambientes e para a aquisição e integração de novas experiências têm que ser desenvolvidos.

2 Objetivos da Tese

O objetivo deste trabalho é aprimorar a efetividade e a eficiência do planejamento de programas de mensuração baseados em GQM através da reutilização de conhecimento baseado em experiência em mensuração sob forma de lições aprendidas em uma empresa de software em programas de mensuração anteriores. Em função destes objetivos, uma metodologia é desenvolvida, implementada e avaliada.

O planejamento das atividades dos programas de GQM os quais necessitam de suporte é composto por [GR99,BDR96,GHW95,BDR94b]: Identificação das metas GQM, desenvolvimento de uma plano GQM e desenvolvimento de procedimentos e instrumentos de mensuração (ver Seção 3.1) .

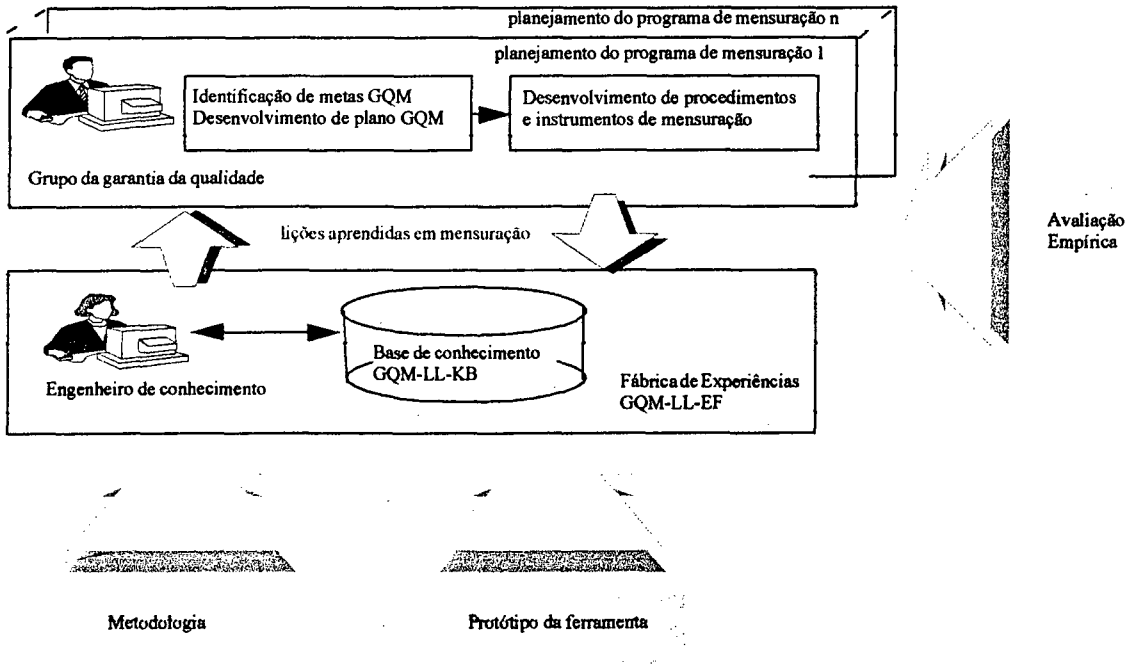


Figura 2 Contribuições da tese

Uma estrutura para a operacionalização da reutilização de lições de mensuração de software é desenvolvida no contexto do *Quality Improvement Paradigm* (QIP) [BCR94a]

aplicado à Fábrica de Experiência (FE) baseado em RBC, referido como Fábrica de Experiência (GQM-LL-EF) e enfoca aspectos técnicos[1]. Esta estrutura modela a representação e armazenamento de lições aprendidas em uma Base de Conhecimento (GQM-LL-KB). Tal estrutura inclui técnicas para recuperação de similaridade de lições aprendidas com o objetivo de apoiar o processo de planejamento de mensuração de software. Técnicas para a contínua aquisição e integração de conhecimento são desenvolvidas, definindo como adquirir novas lições de mensuração em projetos de software e como integrar essas lições em uma Base de Experiências existente.

Uma ferramenta protótipo de suporte para a estrutura é desenvolvida para facilitar sua aplicação na prática. A implementação é integrada ao sistema REMEX [Gre99], um ambiente de desenvolvimento de programas de mensuração no desenvolvimento de software que suporta a edição de produtos GQM durante o processo de planejamento GQM.

A efetividade e a eficiência dessa estrutura para o planejamento de mensuração são avaliadas através de um estudo empírico.

1. Aspectos administrativos estão além do escopo deste trabalho de pesquisa.

3 Cenários de Aplicação

Esta seção apresenta cenários de aplicação ilustrando como a reutilização de conhecimento experimental em mensuração pode apoiar e melhorar o planejamento de programas de mensuração baseados em GQM.

3.1 Abordagem *Goal/Question/Metric*

A abordagem *Goal/Question/Metric* (GQM) [BDR96,BCR94b,BW84] é uma tecnologia específica para mensuração orientada a metas em projetos de software. A tarefa de análise de um programa de mensuração baseado em GQM tem que ser especificada precisamente e explícita por uma meta detalhada para mensuração. As medidas são derivadas no sentido *top-down* baseadas em metas através de um conjunto de questões e modelos de qualidade. O refinamento é precisamente documentado em um plano GQM, fornecendo um fundamento lógico para a seleção de medidas. Os dados coletados são interpretados em sentido *bottom-up* no contexto de metas GQM, questões e modelos, considerando limitações e concepções principais de cada medida.

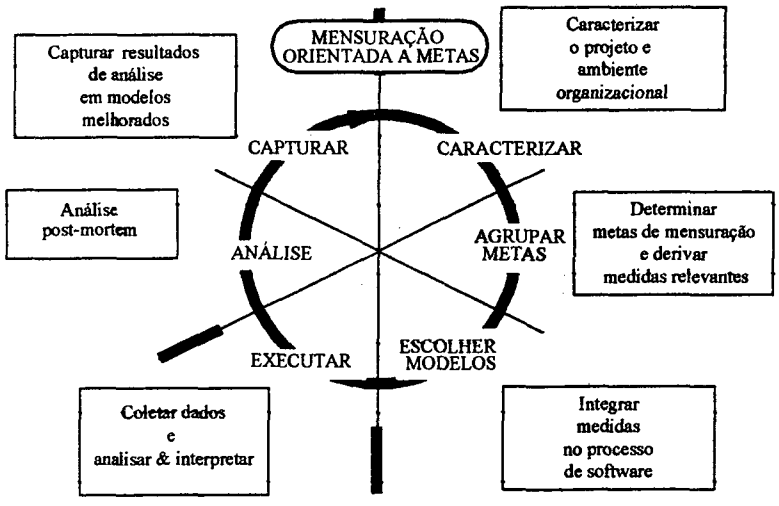


Figura 3 Fases principais do processo GQM

O método GQM descreve o planejamento e o processo de execução de um programa de mensuração baseado no paradigma GQM. O enfoque é apoiado pelo modelo de processos GQM [GRR98,GRR96,GHW95] o qual guia o planejamento e a execução de um programa de mensuração baseado em GQM. Uma visão das etapas de processo de um programa de mensuração baseado em GQM é dada pela Figura 3 :

GQM1 - Estudo prévio. No início de um programa de mensuração um estudo prévio é realizado para identificar as metas de negócio e de aprimoramento para a organização e para caracterizar o ambiente específico. Um projeto piloto para a introdução do programa de mensuração é selecionado e caracterizado. Pré-requisitos necessários para o estabelecimento da mensuração são completados (ex. modelo de processos de software), bem como pessoas envolvidas no programa são motivadas e treinadas.

GQM2 - Identificação de metas GQM e desenvolvimento de plano GQM. Baseadas na caracterização do ambiente de mensuração, as metas do programa de mensuração são explicitamente definidas em termos de objeto, propósito, foco de qualidade, ponto de vista e contexto. As metas de mensuração são refinadas em um conjunto de medidas relevantes através de questões e modelos de qualidade, resultando em um plano GQM.

GQM3 - Desenvolvimento de um plano de mensuração. Um plano de mensuração é desenvolvido integrando as medidas do plano GQM em um processo de desenvolvimento do projeto de software em estudo. Para as medidas definidas no plano GQM, procedimentos de mensuração são criados, declarando para cada medida quando, como e por quem o dado pode ser coletado. Instrumentos de coleta de dados (ex. ferramentas ou questionários) são desenvolvidos.

GQM4 - Coleta de dados e análise e interpretação. Durante a fase de execução do programa de mensuração os dados são coletados de acordo com os procedimentos de coleta de dados especificados no plano de mensuração. Os dados coletados são validados e armazenados. Seguindo o plano GQM no sentido *bottom-up*, os dados de mensuração coletados são analisados e interpretados em sessões envolvendo a equipe de projeto de software. Baseadas nos resultados adquiridos na avaliação, ações de melhoramentos são iniciadas.

GQM5 - Análise post-mortem. Os resultados da mensuração incluindo os dados coletados e sua interpretação são analisados após a conclusão do projeto. O conhecimento obtido durante a mensuração é formulado em relação a fatores de contexto relevantes.

GQM6 - Armazenamento. Os resultados da análise e conclusões são reunidos em modelos organizacionais aperfeiçoados.

O processo de planejamento do programa de mensuração baseado em GQM compreende as etapas GQM1 até GQM3. No contexto deste trabalho, são apresentadas especialmente nas fases principais do processo de planejamento as etapas GQM2 e GQM3. Uma descrição detalhada da fase de planejamento da abordagem GQM é mostrada em Seção 5.

3.2 Cenários de reutilização de experiências

A ampla reutilização de lições aprendidas em todas as etapas do processo de planejamento GQM, na empresa pode auxiliar na prevenção da repetição de falhas em programas de mensuração anteriores e guiar a solução de problemas que ocorreram atualmente. A captura sistemática de tais lições aprendidas baseada em experiências obtidas em programas de mensuração específicos a uma organização, facilita a utilização de tecnologias por criar continuamente um corpo de conhecimento baseado em experiência de como aplicar o enfoque GQM na prática. A reutilização de lições aprendidas em mensuração aborda basicamente dois objetivos: advertir quanto a possíveis falhas e guiar a solução de problemas. Durante cada passo do processo de planejamento da mensuração GQM, lições aprendidas em mensuração podem ser reutilizadas para esses dois objetivos. Os seguintes cenários ilustram como esse conhecimento pode ser reutilizado com o objetivo de apoiar o enfoque GQM.

Supondo uma companhia, IntelliCar, que produz software embutido para automóveis, tenha dois departamentos: FI que desenvolve software para dispositivos de injeção de combustível e ABS que desenvolve software para controle de dispositivos de freio ABS. Como a empresa produz software embutido, um dos mais importantes objetivos é produzir software com zero defeitos. Com esse objetivo, o departamento FI estabeleceu um bem

sucedido programa de melhoria da qualidade baseado em mensuração há dois anos atrás. Agora, o departamento ABS também quer iniciar a utilização de mensuração para aperfeiçoamento. Como o contexto dos dois departamentos são semelhantes e a meta para melhoria é a mesma, experiências disponíveis no departamento FI podem ser reutilizadas pelo ABS com o objetivo de reduzir o esforço de planejamento e melhorar a qualidade do programa de mensuração.

3.2.1. Cenário 1: Aviso de potenciais falhas

As tarefas do processo GQM são complexas e conseqüentemente a propensão a erros é alta. Muitos problemas críticos podem ocorrer durante sua aplicação na prática. Muitos problemas poderiam ser prevenidos, se seu potencial pudesse ser conhecido antecipadamente pelo responsável pela mensuração. Conseqüentemente, uma visão de todas as experiências disponíveis sobre problemas em uma tarefa GQM em particular que ocorreu no passado pode indicar possíveis problemas ao responsável pela mensuração antes da execução das tarefas de planejamento da mensuração GQM. Por exemplo, durante o desenvolvimento do plano GQM, entrevistas são realizadas com o objetivo de adquirir todas as informações importantes para as metas do GQM. Antes de começar tais entrevistas, o engenheiro de garantia de qualidade, ainda sem experiência, responsável pelo estabelecimento do programa de mensuração no departamento ABS pode solicitar experiências adquiridas em problemas que ocorreram quando esta mesma tarefa foi executada em programas de mensuração passados. Por exemplo, o departamento FI pode ter encontrado o seguinte problema: *o entrevistado não forneceu qualquer informação, porque ele/ela não sabia quais eram os objetivos do programa de mensuração e quais informações eram esperadas dele/dela.* O conhecimento dos problemas ocorridos nesta tarefa no passado, irá alertar o responsável sobre possíveis problemas, tentando prevenir sua repetição. Algumas vezes, problemas que ocorrem em fases subsequentes do programa de mensuração são devidos a falhas que aconteceram em fases anteriores. Por exemplo: *O desenvolvimento do plano GQM foi complicado, porque somente conhecimento incompleto foi adquirido durante as entrevistas, ex. as categorias de classificação (nenhum, médio, alto, experiente) do fator de contexto "experiência de desenvolvedores" não foi definido. Conseqüentemente, entrevistas adicionais foram necessárias para definir precisamente as categorias de classificação e sua semântica, ex. o*

nível de experiência é alto se o desenvolvedor trabalhou por mais de dois anos em desenvolvimento de sistemas de telecomunicação, antes do desenvolvimento do plano GQM pudesse ser continuado. O fornecimento de um conjunto de lições aprendidas disponíveis sobre problemas que foram originados na tarefa GQM de interesse do responsável pela mensuração irá apontar possíveis falhas antecipadamente. Adicionalmente, a coleta sistemática de conhecimento experimental além do escopo de um projeto individual irá promover o aprendizado e facilitar o estabelecimento da tecnologia de mensuração de software dentro da empresa.

3.2.2. Cenário 2: Guia da solução de um problema

Quando um problema ocorre durante a execução da tarefa GQM, sua solução pode ser guiada através do conhecimento experimental sobre problemas similares e de como foram resolvidos em programas de mensuração passados. Um exemplo de um problema que aconteceu durante o desenvolvimento de modelos de qualidade no departamento ABS pode ser: *O modelo de qualidade da distribuição de defeitos por criticalidade não pode ser completamente definido se as categorias para defeitos por criticalidade são desconhecidas.* Quando se possui a possibilidade de encontrar com facilidade experiências anteriores a partir de uma descrição do contexto atual do problema a ser enfrentado, um caminho mais rápido e seguro para uma solução pode ser obtido. Baseadas em características de contexto da situação do problema a na sua descrição (ver acima), lições de mensuração relevantes são obtidas da Base de Experiência e fornecidas ao responsável pela mensuração. Além das características e da descrição do problema, a causa do problema é também importante para a seleção de uma estratégia de solução adequada Assim, as causas de problemas passados que foram explicitamente capturadas no passado são também fornecidas. Considerando essas informações, o engenheiro de garantia de qualidade pode explorar a reutilização de sugeridas candidatas a solução com o objetivo de selecionar aquelas que melhor ajustam-se a reais necessidades e características. Por exemplo, a causa de um problema similar em uma base de experiências foi: *Devido à aquisição incompleta de conhecimento em entrevistas, a informação sobre a definição de categorias estava faltando.* Se o problema atual foi causado por uma razão similar, o conhecimento de como o problema foi resolvido no passado, p.ex. *foram realizadas entrevistas com entrevistados que forneceram uma válida categorização de*

defeitos por criticalidade, pode guiar a solução do problema atual. Uma descrição explícita dos resultados obtidos pela aplicação de uma solução em problemas passados, pode mais adiante indicar o que foi melhor para a resolução do problema e o que não foi, em um ambiente em particular. Ex. *que a repetição das entrevistas serviu como base para a definição do modelo de qualidade e que isso contribuiu para a resolução do problema.* Além de reunir estratégias de solução bem sucedidas, recuperar também experiências relativas a tentativas sem êxito para resolver problemas, pode fornecer informações adicionais sobre possíveis falhas e auxiliar a prevenir a aplicação de soluções ineficazes no futuro.

4 Requisitos

Este capítulo descreve alguns requisitos importantes para uma plataforma integrada de suporte, que possa operacionalizar a reutilização e o aprendizado a partir de lições aprendidas em mensuração baseadas na abordagem da Fábrica de Experiência.

A lista de requisitos é baseada nas necessidades específicas para o suporte baseado em experiências para o planejamento de programas de mensuração [GAB00,GAB99,Gre00, GAB+98,GMA+98,GWB98,GB97a]. Este suporte é baseado nas nossas experiências utilizadas neste trabalho através da aplicação da mensuração baseada em GQM aplicada em pesquisa e em projetos de transferência de tecnologia na indústria [GW00,CFF+98,GU97,KL96,CEM96] concentrando a reutilização no domínio da Engenharia de Software [ABT98,Gre98,Ruh98, ABH+97, Hen97, BCR94a,BR91] :

REQ1 Recuperação orientada a metas. Como ilustrado nos cenários de aplicação (ver Seção 3) , lições aprendidas em mensuração podem ser reutilizadas para vários objetivos, tais como prevenção de possíveis falhas e guia para soluções de problemas. Além disso, o conhecimento armazenado na base de experiências também pode ser utilizado com o objetivo de identificar padrões que possam produzir um corpo de *know-how* em mensuração de software mais generalizado. Desta forma, para habilitar um suporte global, a FE tem que fornecer suporte para a reutilização de vários cenários. Lições aprendidas em mensuração de software podem ser recuperadas da BE para várias tarefas de processo de software, pontos de vista e ambientes objetivando vários propósitos. Isto requer a definição explícita das metas de recuperação baseadas na reutilização de cenários e em um mecanismo genérico de recuperação, o qual é adaptável e parametrizável a uma meta particular, e supre experiências úteis em relação a uma meta específica..

REQ2 Representação Compreensiva de conhecimento experimental em mensuração de software. O conhecimento experimental em mensuração de software representando com aplicar, utilizar e adaptar o enfoque GQM a um ambiente específico deve ser representado adequadamente e compreensivamente em uma forma reutilizável com o objetivo de permitir sua

reutilização efetiva e eficiente em projetos futuros. Além disso, interdependências entre partes das lições aprendidas (ex. problemas, soluções e conseqüências) têm que ser representadas adequadamente, de modo que permita que usuário descubra a siga relacionamentos e dependências.

REQ3 Caracterização e organização de experiências. Com o objetivo de permitir uma reutilização eficiente e inteligente, lições aprendidas têm que ser associadas a uma caracterização apropriada e não ambígua do ambiente do qual tiveram origem. Essa descrição é necessária, pois habilita a identificação de experiências adequadas a um projeto específico. No entanto, tais índices variam entre organizações e em relação ao tempo. Assim, um método de aquisição tem que ser elaborado, definindo o processo de como identificar o conjunto mínimo de características de contexto relevantes e determinantes (ex. domínio de aplicação, linguagem de programação) para ser associado às lições aprendidas usadas como índices para a recuperação em um ambiente específico. Portanto, as dependências entre as características de contexto e as lições aprendidas tem que ser investigadas e modeladas explicitamente. Como o domínio da Engenharia de Software é caracterizado por freqüentes mudanças, o contínuo aprendizado deve ocorrer para melhorar e adaptar os índices a ambientes específicos.

REQ4 Recuperação baseada em similaridade de lições relevantes aprendidas baseada em informações incompletas e inconsistentes. Muitos aspectos do domínio de Engenharia de Software não são bem compreendidos atualmente. Assim, o conhecimento armazenado na BE pode ser incompleto ou pode ser impossível declarar todas as características necessárias da situação atual como entrada ao pedido de recuperação. Além disso, não existe uma terminologia geral para os termos de Engenharia de Software. Portanto, mesmo dentro de uma organização, freqüentemente conceitos divergentes, modelos e termos são inconsistentemente usados para descrever aspectos de processos e produtos de software. Outro problema, é que cada projeto de software e cada programa de mensuração são diferentes, e não é muito fácil encontrar um artefato na Base de Experiências que represente as necessidades da situação atual completamente. Como a usabilidade das experiências passadas pode somente ser determinada quando tenta-se reutilizá-las no projeto, o critério posterior de usabilidade é reduzido a priori ao critério de similaridade das experiências, assumindo que situações similares necessitam soluções semelhantes. Isso não é uma tarefa trivial. Como tal atividade considera a avaliação e comparação de representações

complexas de conhecimento com o objetivo de definir similaridade relativa a reutilização de experiências em mensuração.

REQ5 Aprendizado progressivo e contínuo a partir de experiências individuais em projetos. Hoje, empiricamente validado, o *know-how* geral em mensuração de software é insuficiente. Portanto, este *know-how* tem que ser continuamente acumulado através da aquisição de novas experiências, cada vez que um programa de mensuração de software é planejado e executado. A evolução crescente da Base de Experiências necessita uma contínua aquisição e garantia de qualidade dessas novas experiências, bem como sua integração na Base de Experiências existente. Além disso, experiências concretas de projetos de software particulares podem ser generalizadas em padrões genéricos através de empresas para desenvolver um corpo aceito por toda a empresa de *know-how* em mensuração.

Tais necessidades mostram que suporte sofisticado é obrigatório para a operacionalização da abordagem da Fábrica de Experiência permitindo a efetiva e eficiente reutilização de lições em mensuração de software aprendidas durante o processo de planejamento da mensuração.

5 Planejamento de Programas de Mensuração Baseados em GQM

Este trabalho de pesquisa enfoca o suporte baseado em experiências para o planejamento de um programa de mensuração. Desse modo, principalmente as seguintes tarefas de processo são consideradas:

Identificação das metas de mensuração, desenvolvimento de planos GQM, e desenvolvimento de planos de mensuração. Essas tarefas são descritas em detalhes nos capítulos seguintes seguindo o enfoque GQM-R+ [Gre00] baseado no modelo de processo GQM [GRR98,GWH95]. Além disso, o impacto de reutilização de lições aprendidas em mensuração é apontado para cada tarefa do processo de planejamento.

5.1 GQM2.1: Identificação das metas GQM

Quando se planeja um plano de mensuração orientado a metas, o primeiro passo é especificar as metas a serem alcançadas pelo programa de mensuração. Dependendo das metas de negócio e aprimoramento das organizações (por exemplo, "reduzir esforços em manutenção") e dos problemas existentes no processo de software (por exemplo, revisões ineficientes), possíveis metas de mensuração têm que ser identificadas com grande cuidado por parte dos responsáveis pelo programa de mensuração. Baseadas em uma descrição informal das metas de mensuração, dimensões das metas GQM têm que ser derivadas. Um modelo [BDR96,BR88] orientando a definição das metas é estruturado como segue:

Dimensão	Definição	Lista de Exemplos	Exemplo
Objeto	O que será analisado?	processos, produtos, recursos	processo de desenvolvimento de software
Propósito	Por que o objeto será analisado?	caracterização, avaliação, monitoramento, previsão, controle, melhoria [BDR96]	caracterização

Tabela 1 Modelo de metas GQM

Foco de Qualidade	Qual a propriedade do objeto será analisada?	custo, correção, remoção de defeitos, mudanças, confiabilidade, amigável ao usuário, manutenibilidade	confiabilidade
Ponto de Vista	Quem utilizará os dados coletados?	usuário, administrador senior, administrador de projeto, desenvolvedor, testador do sistema, gerente de garantia de qualidade	desenvolvedor de software
Contexto	Em que ambiente a análise ocorre?	organização, projeto, problema, processos, etc.	companhia IntelliCar/ABS

Tabela 1 Modelo de metas GQM

A identificação das metas GQM tem que ser realizada cuidadosamente, considerando todas as características, padrões e terminologia do ambiente. Uma conseqüências dessa atividade pode ser o alto consumo de tempo.

Impacto da reutilização

O relacionamento entre as metas de negócio e melhoramento e as possíveis metas de mensuração em consideração geralmente não é muito evidente, especialmente para pessoas sem experiência em mensuração. A reutilização de lições aprendidas em mensuração pode destacar esse assunto apontando como identificar relacionamentos entre metas de negócio e melhoramento e metas de mensuração. Outros assuntos relacionados com a instanciação do modelo de metas GQM e a seleção de metas pode ser apresentada através das lições aprendidas indicando quais problemas e como eles foram resolvidos em programas de mensuração passados.

5.2 GQM2.2: Desenvolvimento de plano GQM

Um plano GQM é desenvolvido baseado nas metas GQM. O plano GQM consiste nos seguintes componentes [Gre00,BDR96,GHW95]:

- uma meta, definindo o assunto, propósito, foco de qualidade, ponto de vista e o contexto

do programa de mensuração (determinado previamente no passo de processo GQM2A);

- modelos formalizando a meta de mensuração definida;
- um conjunto de questões, operacionalizando a meta;
- um conjunto de modelos, especificando como responder as questões;
- um conjunto de medidas, definindo de modo operacional os dados a serem coletados para satisfazer os modelos.

Atividades de apoio ao desenvolvimento das partes do plano GQM são descritas nas seções seguintes.

5.2.1. GQM2.2.1: Formalização da meta

O paradigma GQM combina modelos de um objeto de estudo (por exemplo, processo de software, produto ou recurso) e um ou mais focos de qualidade (por exemplo, confiabilidade ou manutenibilidade), observando o objeto de estudo para características particulares que podem analisar um foco de qualidade específico a partir de um ponto de vista específico em um ambiente particular [Bas99]. A identificação de medidas úteis necessita ser baseada explicitamente e precisamente em modelos definindo os assuntos apontados na meta GQM de interesse. Por exemplo, com o objetivo de derivar medidas para um meta desejada em "analisando o código fonte de um componente com o propósito de predizer sua manutenibilidade a partir da perspectiva do engenheiro de manutenção" necessita-se entender modelos de manutenibilidade (exemplo, quantos componentes mudaram) e o código fonte do próprio componente (por exemplo, aspectos estruturais que podem causar impacto sobre manutenibilidade). Assim, o objeto de estudo e o foco de qualidade da meta GQM são formalizados através de modelos com o objetivo de fornecer uma base explícita para o planejamento futuro de programas de mensuração relativos a um ambiente específico.

A formalização de metas envolve basicamente o desenvolvimento de modelos formalizando o objeto de estudo e o foco de qualidade apontado pela meta GQM de interesse. O objeto de estudo é explicitamente descrito através do modelo do respectivo processo, produto ou recurso no contexto específico, fornecendo um entendimento comum e imparcial do objeto de estudo. Para se refinar a meta em medidas operacionais, o foco de qualidade

necessita ser refinado adequadamente e fatores de influência relevantes necessitam ser identificados. Como o refinamento do foco de qualidade depende do contexto específico no qual a mensuração ocorre, do objeto a ser analisado e do respectivo ponto de vista, o refinamento tem que ser feito levando em consideração tais fatores. O foco de qualidade é modelado na forma de um modelo da qualidade sob ponto de vista determinado na meta no contexto específico. O objetivo do desenvolvimento do modelo de qualidade é sugerir uma definição de alto nível do foco de qualidade, relacionando as questões individuais do plano GQM ao foco de qualidade identificadas na meta GQM e indicando explicitamente suas interdependências. Se não existe ainda um modelo sobre o foco de qualidade no ambiente específico, as informações respectivas são capturadas durante a etapa de aquisição de conhecimento do processo GQM.

Impacto da reutilização

A formalização da meta GQM pode ser apoiada por lições aprendidas considerando o desenvolvimento o modelo de objeto ou de qualidade. Tais lições aprendidas podem apontar, por exemplo, os seguintes problemas possíveis:

- Relacionamentos do foco de qualidade na meta GQM com o modelo de qualidade, por exemplo, se um modelo de qualidade mostrou-se ser inapropriado para um ambiente específico ou para alcançar a meta GQM particular,
- Definições incompletas dos modelos de qualidade devido às variáveis independentes não consideradas ou àquelas consideradas irrelevantes em um ambiente específico,
- Definição incompleta ou inapropriada do modelo de objeto, não capturando compreensivamente ou corretamente o processo de software, produto ou recurso em um ambiente específico.

5.2.2. GQM2.2.2: Aquisição de conhecimento

Para refinar a meta em medidas operacionais, o foco de qualidade necessita ser definido adequadamente e satisfazer as necessidades do ponto de vista. Conhecimento relevante é tanto representado no modelo de qualidade definido na tarefa GQM 2.2.1 como também é adquirido através de entrevistas com o pessoal indicado no ponto de vista da meta GQM. Essa

informação é usada para derivar modelos de qualidade válidos e corretos, para identificar fatores de contexto relevantes, e então medidas relevantes. Durante as entrevistas, os seguintes tópicos deveriam ser abordados [GRR94] (ver Figura 4):

- **Foco de Qualidade**. É especificado nas entrevistas o que significa o foco de qualidade para os entrevistados. O foco de qualidade é refinado em um conjunto de fatores de qualidade.
- **Hipótese inicial**. Para cada fator de qualidade relativo ao foco de qualidade, uma suposta distribuição de valores pode ser determinada, baseada na intuição do entrevistado e no conhecimento de domínio disponível.
- **Fatores de variação**. Os fatores que causam impacto sobre os fatores de qualidade são declarados.
- **Impacto na hipótese-inicial**. Para cada fator de variação, o impacto esperado do fator de variação no fator de qualidade deve ser especificado, quando possível.

Um instrumento comumente utilizado para a aquisição e estruturação do conhecimento durante as entrevistas é folha de abstração [GRR94]. Uma folha de abstração é um documento de uma página formado por quatro quadrantes, um para cada um dos tópicos mencionados, com a respectica meta GQM no cabeçalho (ver exemplo na Figura 4).

META	processo de desenvolvimento de sw	caracterização	confiabilidade	desenvolvedor	IntelliCar/ABS
Foco de Qualidade 1 no. total de defeitos 2 no. de defeitos por criticalidade (não crítico, critico, outro) 3 no. de defeitos por fase de detecção (REQ, HLD, LLD/IMP) 4 total de esforço de retrabalho (em horas)			**Fatores de Variação** Adaptação do processo: 1 tipo de inspeção de código Domínio de conhecimento: 2 experiências dos membros da equipe de desenvolvimento		
Hipótese Inicial 1 no. total de defeitos: 100 2 75% não crítico, 25% crítico 0% outros; 3 REQ 20, HLD 20, LLD/IMP 40 4 total de esforço de retrabalho: 1000 h			**Impacto na Hipótese Inicial** Adaptação do processo: 1 inspeções de código ad-hoc são menos efetivas que outro tipos da inspeção Domínio de conhecimento 2 membros da equipe de desenvolvimento com mais experiência introduzem menos defeitos		

Figura 4 Folha de abstração - Exemplo simplificado

Impacto da reutilização

A aquisição de conhecimento durante as entrevistas é uma fase bastante crítica para a qualidade do programa de mensuração desenvolvido. Entrevistadores sem experiência podem cometer muito erros resultando em definição de medidas inadequadas e irrelevantes. Assim, a indicação de possíveis problemas que podem ocorrer durante a definição de questões, modelos e medidas baseados no conhecimento obtido em entrevistas podem fornecer suporte significante ao processo de aquisição de conhecimento. Lições aprendidas considerando a execução de entrevistas (ex., duração, preparação, material necessário, motivação do entrevistado, etc.) podem auxiliar na realização bem sucedida de entrevistas. Por exemplo. antes de executar entrevistas, o entrevistador pode questionar possíveis problemas que podem acontecer durante uma entrevista antecipadamente (ex., entrevistado não quer fornecer as informações necessárias devido a motivação insuficiente) visando prevenir sua repetição. Além disso, lições aprendidas podem guiar a solução de problemas atualmente mostrando estratégias de solução aplicadas no passado (ex., objetivo da entrevista foi explicado em detalhes e confirmada a confidencialidade relativa as informações adquiridas).

5.2.3. GQM2.2.3: Desenvolvimento de questões GQM

O plano GQM é desenvolvido baseado em parte na definição do modelo de qualidade em GQM2.2.1 e no conhecimento adquirido durante as entrevistas em GQM2.2.2. Para cada fator de qualidade e fator de variação documentado na folha de abstração, uma questão no plano GQM é refinada, expressando uma necessidade de informações [BDR96,GHW95]. As questões representam um refinamento intuitivo da meta de mensuração. Uma hipótese pode visualizar uma questão e pode ser declarada explicitamente baseada em hipóteses dadas na folha de abstração. A hipótese especifica a distribuição esperada de valores relativos a fatores de qualidade e fatores de variação concentrado na pergunta no ambiente específico. Como planos GQM consistem geralmente de grande números de questões, categorias de questões [BDR96,BR88] (ver Tabela 2) foram definidas com objetivo de guiar sua derivação e estruturar planos GQM. Continuando o exemplo acima, questões são ilustradas na Tabela 3.

Categoria	Descrição
Foco de Qualidade	questões envolvendo o(s) modelo(s) de qualidade a serem usados que definem também o foco de qualidade declarado na meta.
Processo/Definição do Produto	questões envolvendo fatores que podem ter impacto sobre os valores dos modelos de qualidade. Dependendo se o objeto de estudo é um processo, produto, essa categoria é referenciada tanto como definição de produto ou de processo.
Definição do Processo - Adaptação do processo	questões empreendendo a captura de informações envolvendo a conformidade do processo organizacional atual ao processo oficial
Definição do Processo - Domínio de entendimento	questões envolvendo os atributos de objetos usados pelo processo em estudo e os atores realizando o processo.
Definição do produto	questões envolvendo atributos lógicos e físicos do produto, custo de desenvolvimento relacionado ao produto, mudanças no produto e contexto operacional do produto.

Tabela 2 Categorias de questões

Impacto de reutilização

Derivar um conjunto completo de questões formuladas precisamente a partir folha de abstração ou do modelo de qualidade é difícil. Essa tarefa pode ser apoiada fornecendo-se linhas mestras em como formular questões baseadas no conhecimento adquirido considerando-se a meta de mensuração e o contexto específico baseado nas informações adquiridas na folha de abstração.

5.2.4. GQM2.2.4: Desenvolvimento de modelos de qualidade

Cada questão no plano GQM é formalizada pela definição detalhada dos modelos de qualidade [Gre00,GB98,BDR96] (ver Tabela 3). Esses modelos fornecem meios para responder as questões. Modelos operacionalizam as questões do plano GQM, quantificam os vários atributos abstratos dos artefatos em estudo e definem precisamente como comparações de qualidade/produtividade, avaliações e previsões devem ser feitas [BDR96]. Os modelos são freqüentemente desenvolvidos considerando modelos abstratos, exemplo, tamanho, que tem que ser refinado em modelos operacionais descritivos, conduzindo a mensuração. Modelos devem ser cuidadosamente analisados com o objetivo de determinar a validade de suas premissas e sua aplicabilidade em um ambiente particular em estudo. Baseados na definição de modelos, atributos relevantes a serem medidos com o objetivo de satisfazer os modelos são

Programa de Mensuração da IntelliCar/ABS
Meta GQM
Analisar o processo de desenvolvimento de software com o objetivo de melhorar a confiabilidade sob o ponto de vista dos desenvolvedores de software da IntelliCar/ABS
Questões GQM
Definição de Qualidade Q1. Qual é o número total de defeitos detectados antes da entrega do produto? Q2. Qual a distribuição de defeitos relatada antes da entrega por criticalidade ? Q3. Qual o número de defeitos por fase de ciclo de vida de detecção? Q4. Qual o esforço total de retrabalho ? Definição do Processo Q5. O tipo de inspeções tem impacto sobre a efetividade das inspeções? Q_5.1. O que é a efetividade de inspeções? Q_5.2. Quais os tipos de inspeções são aplicadas? Q6. A experiência dos desenvolvedores tem impacto sobre o número de defeitos apresentados pelo sistema? Q_6.1. Qual o nível de experiência dos membros da equipe de desenvolvimento? Q_6.2. Qual o número de falhas detectadas antes da entrega ?
Modelos de Qualidade
Modelo 5.1 Efetividade das Inspeções *Contexto*: Empresa IntelliCar, departamento ABS *Hipótese* A densidade de defeitos é comparável através dos documentos. *Descrição do modelo:* Efetividade = (número de defeitos detectado durante as inspeções)/(tamanho do documento* duração do treinamento) *Atributos*: Número de defeitos detectados durante as inspeções; tamanho dos documentos; duração do treinamento ...
Medidas [nível de mensuração: intervalo]
M1. cálculo das falhas relatadas antes da entrega [razão: inteiro] M2.1. para cada defeito relatado antes da entrega: classificação por criticalidade [ordinal: {não crítico, crítico, outros}] M2.2. cálculo das falhas relatadas antes da entrega [razão: inteiro] M3.1. para cada defeito relatado antes da entrega: fase de ciclo de vida de detecção [nominal: {REQ, HLD, LLD/IMP}] M4.1. para cada defeito relatado antes da entrega: esforço para isolar as falhas (pessoa-horas) [razão: inteiro] M4.2. para cada defeito detectado antes da entrega: esforço em corrigir a falha (pessoa-horas) [razão: inteiro] M5.1.1. para cada inspeção: cálculo dos defeitos detectados [razão: inteiro] M5.1.2. para cada inspeção: número de operações definidas[razão: inteiro] M5.1.3. para cada inspeção: duração do treinamento (pessoa-horas) [razão: inteiro] M5.2.1. para cada inspeção: tipo de inspeção [nominal: {*ad-hoc*, *checklist* , baseado em cenário}] M6.1.1. para cada desenvolvedor: nível de experiência [ordinal: {nenhum, médio, alto, especialista}] M6.1.2. número de desenvolvedores [razão: inteiro] M6.2. cálculo das falhas relatadas antes da entrega [taxa: inteiro]

Tabela 3 Plano GQM - exemplo simplificado

determinados.

Impacto da reutilização

Os modelos devem ser desenvolvidos levando em conta o ambiente especificado, visto que eles geralmente tem que fazer hipóteses simplificadas. Então, características, padrões e terminologias têm que ser conhecidas. Desenvolver tais modelos a partir de cada questão do plano GQM é uma tarefa intelectual complexa e necessita uma grande quantidade de esforço. Sugerindo lições aprendidas sobre:

- como esses modelos foram derivados em programas de mensuração passados,
- indicando os aspectos importantes a serem abordados pelos modelos e
- dificuldades encontradas na definição e utilização dos modelos

pode fornecer um sustentável suporte ao desenvolvimento de modelos de qualidade especialmente quando desenvolvidos por funcionários sem experiência em garantia de qualidade.

5.2.5. GQM2.2.5: Desenvolvimento de medidas

As questões do plano GQM são refinadas quantitativamente em um conjunto de medidas através de modelos de qualidade [BDR96,GWH95,BCR94b]. Para cada atributo a ser medido com o objetivo de satisfazer o modelo de qualidade, uma medida é definida. Além disso, o nível de mensuração, a unidade (para medidas numéricas) e faixa dos respectivos dados a serem coletados é definido para cada medida. Continuando o exemplo acima, a Tabela 3 apresenta alguns exemplos de medidas correspondentes.

Impacto da reutilização

A seleção das medidas válidas é um dos mais complicados assuntos na mensuração de software. Assim, a definição das medidas pode ser fortemente apoiada pelas lições aprendidas indicando como medir atributos específicos, como selecionar escalas de mensuração e quais problemas podem ocorrer ou serem causados pelo desenvolvimento de medidas mal definidas.

5.3 GQM3: Desenvolvimento do Plano de Mensuração

O principal foco do plano de mensuração é a integração apropriada da mensuração no processo de desenvolvimento de software em um ambiente específico. Isso inclui a realização de procedimentos de mensuração e instrumentos para a coleta de dados.

5.3.1. GQM3.1: Desenvolvimento de procedimentos de mensuração

Procedimentos de mensuração são definidos pela determinação para cada medida identificada no planos GQM, quando, como e por quem os dados foram coletados [BDR96,GHW95]. Mais detalhadamente, as seguintes atividades devem ser realizadas.
Primeiro, devem ser definidos, considerando o processo de software, os dados a serem coletados. Três tipos principais de estratégias podem ser identificadas [BDR96]: periodicamente (por exemplo, semanalmente), início/fim das atividades/fases do processo de software (por exemplo, fim da análise de requisitos), ou quando um artefato alcançou determinado estado (por exemplo, finalização da versão 1.0).

Para cada medida, as ferramentas ou pessoas que podem possivelmente fornecer os dados devem ser identificados. Fatores de decisão são: se podem ser coletados automaticamente por ferramentas ou qual função ou posição na estrutura organizacional é melhor adaptada (em termos de confiabilidade, custo, etc.) para a coleta de dados, por exemplo, administrador de projeto, testador ou desenvolvedor. Além disso, deve ser determinado quem é responsável pela garantia da qualidade e pela manipulação/ armazenamento dos dados.

O próximo passo é a especificação dos instrumentos de coleta de dados, definindo como os dados devem ser coletados para cada medida. Três categorias principais de instrumentos de mensuração podem ser identificados [BDR96]: ferramentas (por exemplo, analisadores estáticos de código), questionários (por exemplo, formularios NASA SEL [NAS94]) e entrevistas estruturadas. Um pequeno resumo de um plano de mensuração apresentado na Tabela 4.

Nome	Entidade/ Atributo	Periodicidade	Evento	Recurso/ Coletor de Dados	Responsável pela garantia da qualidade	Instrumento de coleta de dados
esforço de projeto	projeto/ esforço sw	periódico	semanalmente	Humano/ desenvolvedor	Gerente de projeto	Relatório de Esforço
criticalidade do defeito	defeito/ criticalidade	fim do processo	manipulação de defeito	Humano/ testador	Equipe de garantia da qualidade	Relatório de Problema
fase de detecção de defeito	fase defeito/ detecção	fim do processo	manipulação de defeito	Humano/ desenvolvedor	Equipe de garantia da qualidade	Relatório de Problema

Tabela 4 Plano de mensuração - exemplo simplificado

Impacto da reutilização

A definição do plano de mensuração é um processo difícil. Por exemplo, muitas decisões devem ser tomadas em relação aos procedimentos, instrumentos, etc. em dependência ao processo de software organizacional. Essa atividade pode ser apoiada por lições aprendidas indicando o que funcionou no passado e o que não. Por exemplo, lições aprendidas indicando procedimentos de mensuração que se mostraram inapropriados com respeito ao modelo organizacional de processo de software podem prevenir a coleta de dados inválidos no futuro.

5.3.2. GQM3.2: Desenvolvimento de instrumentos de mensuração

Para implementar o plano de mensuração, é necessário desenvolver instrumentos de coleta de dados. Dependendo do tipo dos instrumentos de coleta de dados, tanto uma ferramenta tem que ser desenvolvida, questionários tem que ser elaborados e também entrevistas estruturadas devem ser planejadas. Por exemplo, considerando questionários, a confiabilidade dos dados, bem como sua usabilidade tem que ser asseguradas. Com o objetivo de coletar dados confiáveis, as questões tem que ser cuidadosamente formuladas, todos os termos desconhecidos explicados e as questões abertas limitadas, tanto quanto possível. Usabilidade tem que ser alcançada como o objetivo de reduzir os esforços relacionados com a coleta de dados. Então o questionário deve ser bem estruturado, apresentando questões em uma ordem lógica, etc. Como os exemplos mostraram, os instrumentos de coleta de dados devem ser adaptados ao ambiente de desenvolvimento que requer um conhecimento detalhado da estrutura organizacional, do fluxo de trabalho, dos padrões e da terminologia.

Impacto da reutilização

Lições aprendidas considerando problemas que ocorreram durante o desenvolvimento de instrumentos de mensuração (por exemplo, categorias de respostas do questionário não puderam ser definidas devido à falta de informações no plano GQM ou uma categoria particular estava faltando na coleta de dados passada) pode ajudar a prevenir problemas e também facilitar sua solução (por exemplo, adquirir informações relevantes através de entrevistas adicionais e/ou através da consulta ao modelo de processo de software). A reutilização de lições aprendidas no desenvolvimento de instrumento de mensuração orienta a difícil tarefa de projetar instrumentos apropriados integrados ao processo de software, que pode ser facilmente utilizadas pelos coletores de dados, resultando numa coleta de dados mais completa e válida.

A análise de impacto da reutilização durante cada tarefa do planejamento do processo GQM ilustra o grande potencial da reutilização das lições aprendidas com o objetivo de apoiar o estabelecimento de adaptados programas de mensuração na prática e continuamente melhorar e adaptar o processo de mensuração a características específicas da organização.

6 Mensuração Baseada em GQM: Estado da Arte/Prática

A abordagem *Goal/Questio/Metric* foi desenvolvida em resposta à necessidade de uma abordagem orientada a metas para mensuração processos e produtos em Engenharia de Software. O enfoque GQM foi desenvolvido primeiramente em 1984 na Universidade de Maryland [BW84] e estendido como parte do projeto TAME [BR88]. GQM mostra-se um enfoque em mensuração de software mais flexível e adaptável, suportando o gerenciamento de qualidade e projeto [Cem96,Rom91]. Durante os últimos anos, o enfoque GQM foi melhorado significantemente (ex.,[Gre00,GRR98,BDR96,DHL95]). Representação formal de planos GQM [Dif93,Rom91] e vários modelos para produtos GQM foram desenvolvidos [GRR94,Dif93,Rom91]. Um modelo de processo para planejamento e execução de programas de mensuração foi desenvolvido [Gre00,GRR98,GHW95]. Guias para sua aplicação foram formulados baseados no *feedback* obtido na prática [GHR+97,BDR96,CEM96,NAS94,Rom91]. Custos e benefícios da aplicação do enfoque GQM em projetos industriais foram analisados através de estudos empíricos [GR99,BE97].

A abordagem GQM foi utilizada com sucesso em diversas empresas, como NASA-SEL (EUA) [BCM+92], Robert Bosch GmbH (Alemanha) [BDT96], Allianz Lebensversicherungs-AG (Alemanha) [GRR94], Digital SPA (Itália) [FLM96], Motorola [Das92], Schlumberger (Holanda) [LSO+98,HOL+96,Sol95].

Várias ferramentas foram desenvolvidas para suportar os vários passos do programa de mensuração baseado em GQM. (ex., GQMaspect [PER96], GQMEAPlan [Cin97,Kra97], GQMDiva [Mar95,Fri94,Dif93,Mar93], GQM Tool [FLM+96], ES-TAME [Oiv94,OB92], GQM-Plan [AK98]). Todas as ferramentas disponíveis são protótipos de pesquisa. Não há atualmente software comercial que apóie diretamente a definição de planos de modo *top-down* e interpretação de dados de acordo com os planos.

6.1 GQMaspect

GQM Abstraction Sheet and GQM Plan Editing and Construction Tool (GQMaspect) [PER96] foram desenvolvidos a na Universidade de Kaiserslautern e suporta os artefatos GQM e o processo de desenvolvimento do plano GQM.

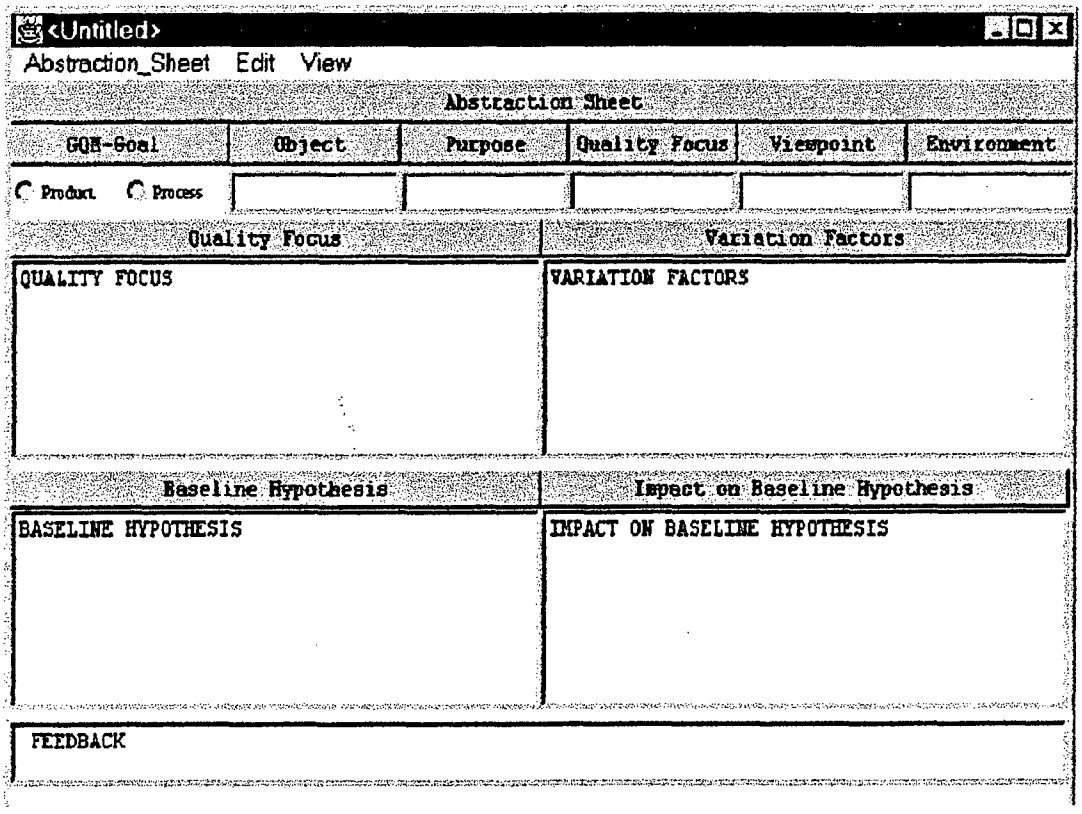


Figura 4 Interface GQMaspect

Suas características mais importantes são:

- definição de metas GQM,
- construção e edição de planos incluindo folhas de abstração, questões e medidas,
- geração de planos GQM a partir de folhas de abstração e vice versa, e
- intercâmbio de folhas de abstração GQM e planos GQM no sistema APEL.

GQMaspect é um protótipo em pesquisa independente de plataforma que executa sobre JAVA. Está disponível publicamente grátis para uso interno em empresas e uso não comercial.

O GQMaspect não suporta o junção de planos GQM em um plano GQM integrado e o desenvolvimento de modelos de qualidade e planos de mensuração.

6.2 GQM-MEAPlan

GQM-MEAPlan [Cin97,Kra97] é um protótipo refinando a ferramenta GQMaspect considerando o desenvolvimento de planos de mensuração e questionarios para a coleta de dados. GQM-MEAPlan é uma ferramenta independente de plataforma implementada em JAVA.

O sistema pode ler planos GQM criados com a ferramenta GQMaspect através de um arquivo de interface definido. GQM-MEAPlan fornece interfaces gráficas de usuário com o objetivo de habilitar a representação e manipulação de procedimentos de mensuração e questionários.

Compreende facilidades como:

- a geração de planos de mensuração baseados em planos GQM pela integração de medidas definidas nos planos GQM em um único plano de mensuração.
- a geração de questionários baseados em procedimentos de mensuração definidos no plano de mensuração e informações adicionais fornecidas pelo usuário no processo de software, produtos ou recursos.

Todos os produtos GQM relatados podem ser editados pelo usuário e permitem impressão em forma de relatórios.

6.3 GQM DIVA

GQM-DIVA [Mar95,Fri94,Dif93,Mar93] é um protótipo de ferramenta de pesquisa desenvolvido na Universidade de Kaiserslautern para a definição, interpretação e validação de planos GQM. É baseado na representação interna de planos GQM consistindo de metas, questões e medidas, que habilitam o armazenamento persistente e manipulação de planos GQM seguindo a definição de linguagem em forma de gramática EBNF [Dif93]. O sistema

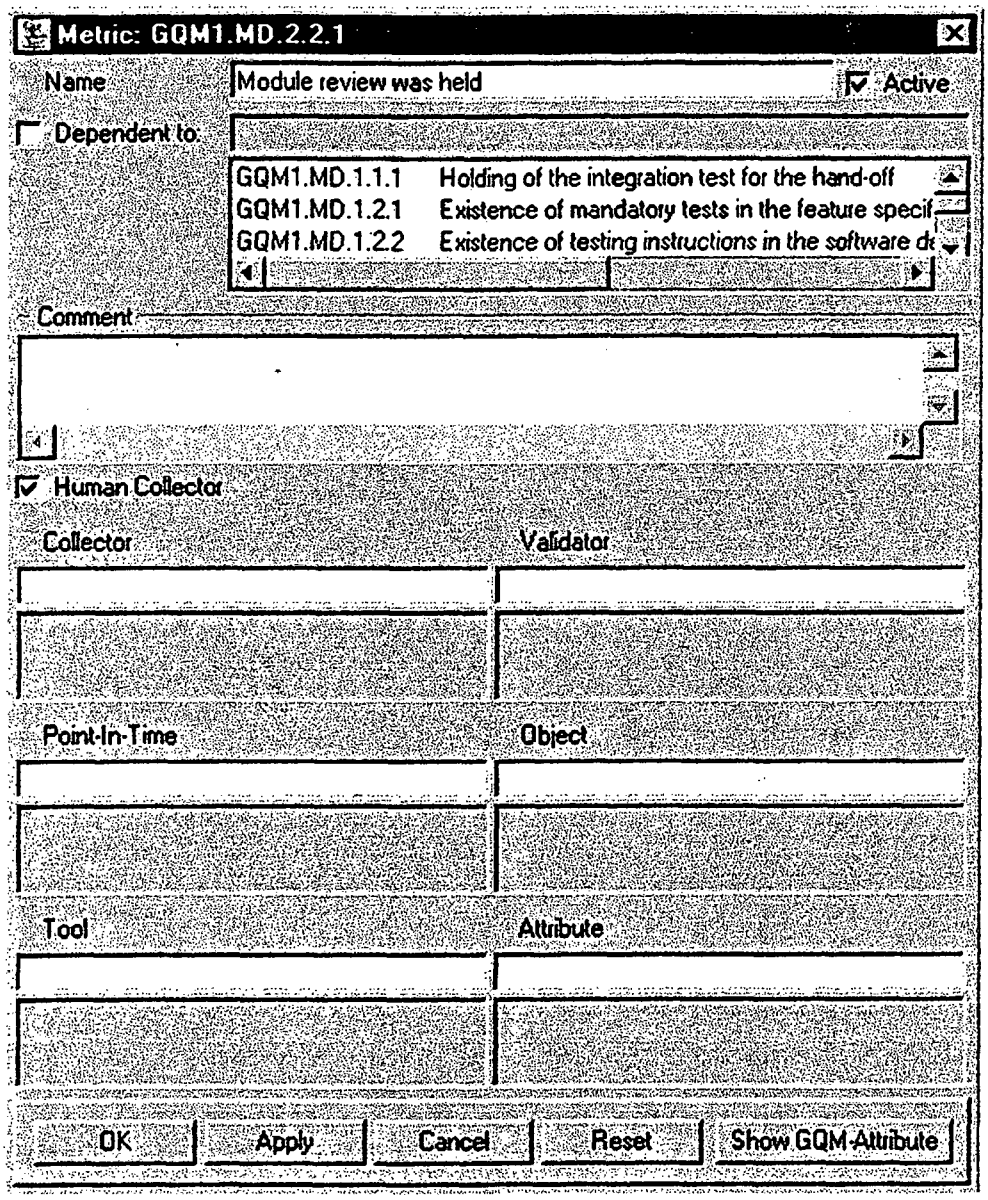


Figura 5 Interface GQMMEAPlan

suporta a criação de planos GQM fornecendo interfaces gráficas e orientados à sintaxe que habilitam a entrada interativa de planos GQM [Mar93]. Esse editor permite que planos GQM sejam criados ou modificados em sua forma textual e armazenados no formato da representação interna. O sistema também incorpora uma ferramenta de análise de planos GQM [Fri94]. Essa ferramenta suporta a análise e validação de planos GQM através de verificação de consistência e semântica e uma visão de hipótese com o objetivo de descobrir questões ou medidas que estejam incorretas ou faltando.

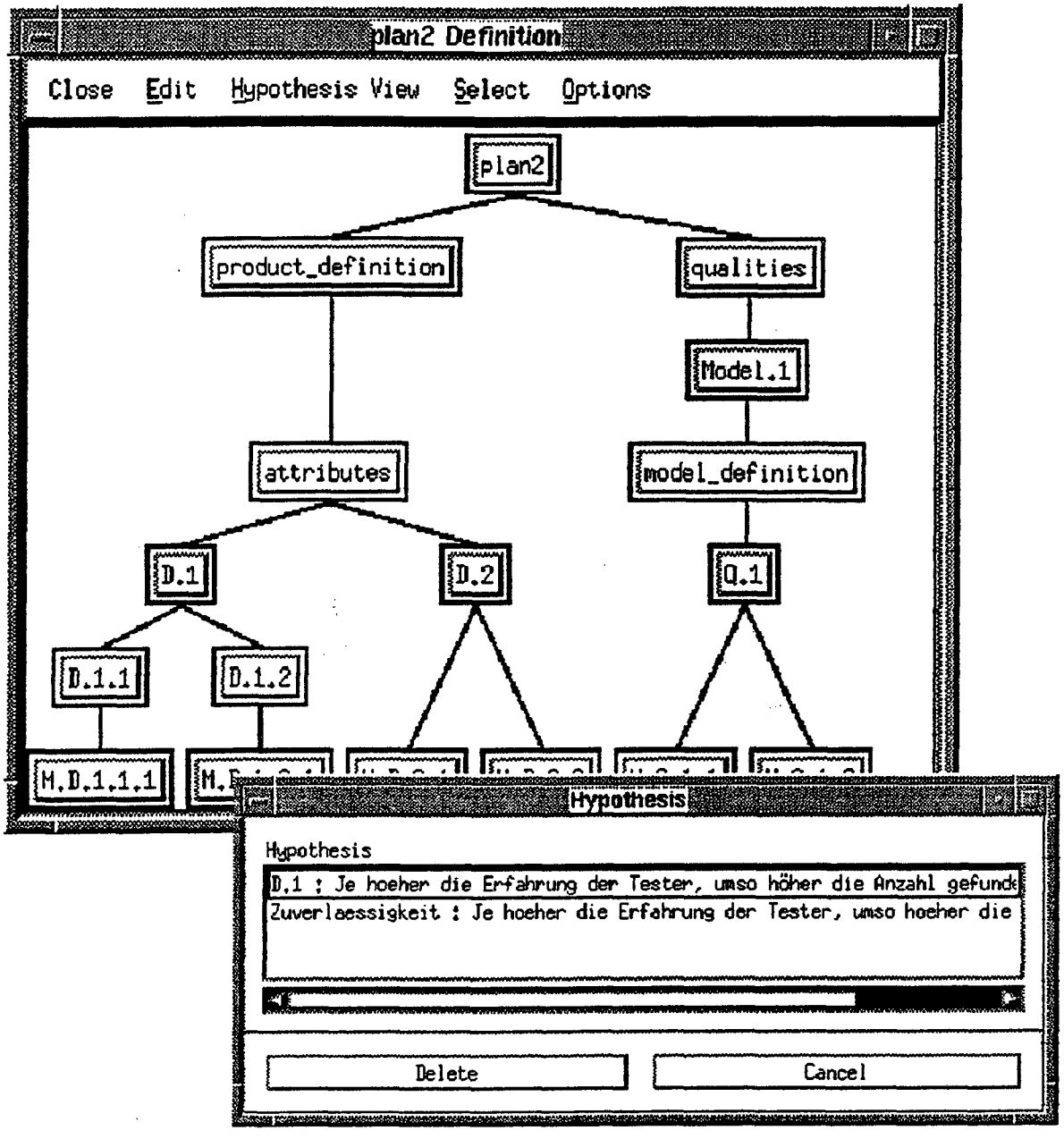


Figura 6 Interface GQM-Diva

GQM-DIVA também suporta a preparação de dados coletados para a análise e interpretação de dados coletados baseados nas hipóteses definidas no plano GQM.

GQM DIVA não considera certos produtos GQM tais como folhas de abstração, modelos de qualidade e plano de mensuração.

6.4 GQM Tool

GQM Tool [FLM+96] foi desenvolvido por CEFRIEL em Milão (Itália). É um sistema que fornece um modo efetivo e rápido de controlar projetos de mensuração baseados na metodologia GQM.

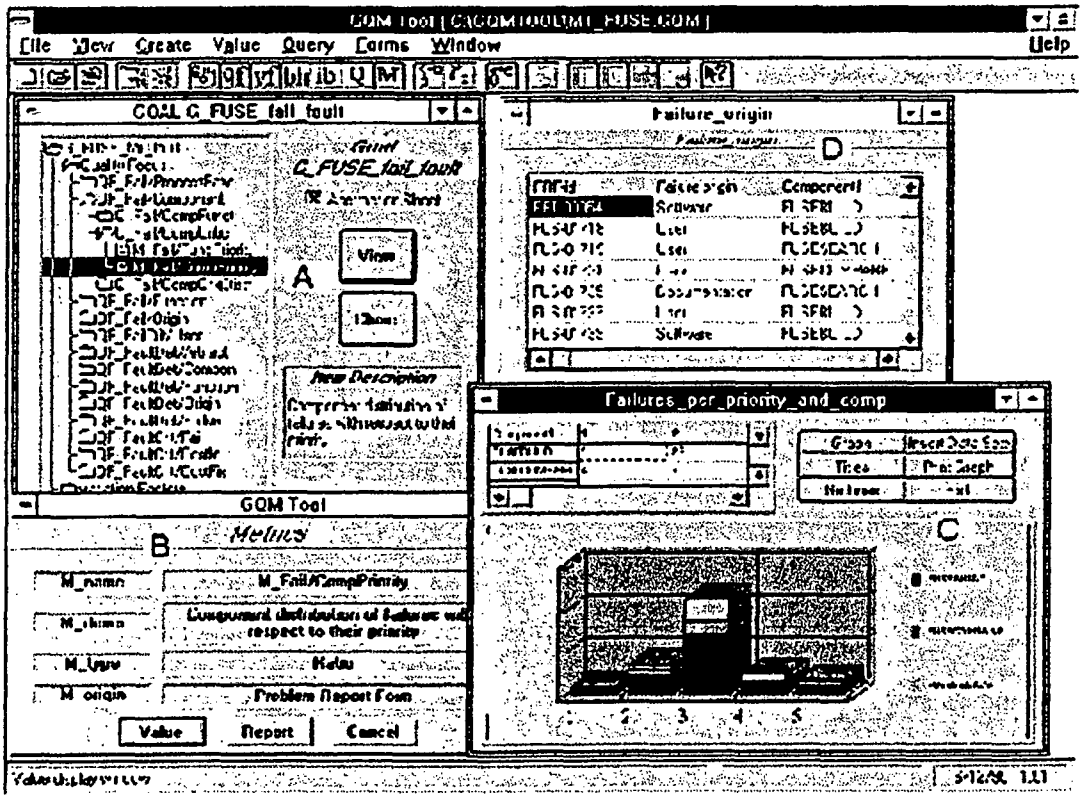


Figura 7 Interface GQM Tool

GQM Tool consiste de quatro seções que permitem:

- Criar e controlar planos GQM.

  Usando formularios de dados, elementos de planos GQM e relacionamentos entre eles podem ser definidos.

- Conectar medidas de planos GQM com bases de dados

  Com GQM Tool pode-ser utilizar os dados contidos em uma base de dados Microsoft Access. A base de dados pode ser consultada e suas tabelas podem ser conectadas com

medidas definidas no plano GQM.

- Definir relatórios gráfico

  GQM Tool fornece uma janela de definição de gráfico que possibilita especificar características gráficas e relatórios gráfico dos conjuntos de dados conectados as medidas GQM.

- Analisar a estrutura GQM inteira

  Um conjunto de comandos e janelas é fornecido e permite que uma análise completa do plano GQM junto com o conjunto de dados.

Além das funcionalidade descritas acima, GQM Tool fornece a possibilidade de produzir e imprimir planos GQM e formulários de coleção de dados. GQM Tool suporta uma visão particular da estrutura do plano GQM composta por quatro níveis de hierarquia: Meta, Folhas de Abstração, Questões e Medidas.

Para criação e edição de planos GQM, CEFRIEL GQM Tool oferece quatro visões, uma para cada quadrante da folhas de abstração. Não apresenta os quadrantes em uma única folha, nem oferece uma visão particular do plano GQM. O desenvolvimento de modelos de qualidade e planos de mensuração não são suportados pela ferramenta.

6.5 ES-TAME

O sistema ES-TAME [Oiv94,OB92] é um protótipo de sistema especialista que suporta os processos de *design* para software embutido de tempo real pela integração de modelos de Engenharia de Software e modelos de qualidade relacionados através de medidas. Uma metodologia suportando o paradigma *Goal/QuestionMetric* é embutido dentro do sistema. O usuário pode construir um plano GQM em forma de metas, questões e medidas utilizando modelos para escrever metas tanto quanto selecionar de uma conjunto pré-definido de questões e medidas ou escrever novas questões e medidas.

A parte sistema especialista do ES-TAME suporta o desenvolvimento de planos GQM usando mecanismos de *forward chaining* para inferir elementos do plano GQM. ES-TAME também pode oferecer automaticamente ao usuário dados sobre os produtos de trabalho para

aquelas medidas para as quais dados existem. O sistema assiste o usuário na resposta a questões baseadas nos dados e na interpretação de resultados em termos de metas.

6.6 GQM-Plan

O protótipo da ferramenta GQM-Plan [AK98] foi desenvolvida na Universidade de São Carlos, no Brasil. O objetivo principal da ferramenta é dar suporte à elaboração do plano GQM. Ele cobre parte do processo de planejamento de um programa de mensuração, incluindo a caracterização do projeto de software, a identificação de metas GQM e o desenvolvimento do plano GQM consistindo da folha de abstração, questões e medidas. O sistema basicamente suporta a edição dos respectivos produtos GQM e gera relatórios documentando o desenvolvimento de produtos GQM.

A ferramenta GQM-PLAN destina-se tanto aos profissionais responsáveis pela realização do programa de qualidade quanto aos membros da empresa que estão envolvidos no programa em questão. A arquitetura básica da ferramenta compreende os módulos de entrada de dados, consistência e emissão de relatórios. No módulo da entrada de dados, os dados necessários para a elaboração do plano GQM são solicitados ao usuário, através dos seguintes sub-módulos para a definição de:

- programa de avaliação,
- metas do programa de avaliação,
- folha de abstração,
- questões, e
- medidas.

O módulo da consistência é responsável por avaliar os dados fornecidos pelo usuário, de forma a garantir que os mesmos estejam corretos, completos e consistentes (p.ex., verificar se um programa de mensuração tem pelo menos uma meta cadastrado).

A ferramenta produz uma série de relatórios referentes a uma ou mais avaliações de qualidade, como p.ex. lista de metas, fatores de variação, relatório das hipóteses básicas, etc.

A ferramenta foi implementada em Delphi 3.0 para funcionar em ambiente PC e sistema operacional Windows.

6.7 Discussão

GQM comprovou ser uma metodologia efetiva e adaptável para o estabelecimento de programas de mensuração de software na prática. Embora tenha sido melhorado consideravelmente durante os últimos anos, mensuração é ainda uma tecnologia nova. Visando desenvolver uma tecnologia madura, experiências sobre sua aplicação na prática tem sido coletadas sistematicamente para construir um corpo de conhecimento em mensuração de software amplo e genéricamente aplicavél. Entretanto, nem o modelo de processo GQM, nem quaisquer das ferramentas suportam sistematicamente a reutilização integrada de experiências em mensuração durante o planejamento ou execução de programas de mensuração. A exceção é o sistema ES-TAME que fornece suporte para modelos de Engenharia de Software reutilizáveis e adaptáveis e uma metodologia estruturada para construção de planos. Porém, o ES-TAME cobre mais o desenvolvimento de planos, ao invés do processo de planejamento completo e seus produtos relacionados (por exemplo, de acordo com [GRR98, BDR96,GHW95,GRR94]). O sistema também oferece facilidades de procura e navegação sem suportar a recuperação de planos GQM similares. Como o sistema é baseado em uma máquina de inferência baseada em regras a manutenção do sistema se torna complicada, fazendo com que o aprendizado a partir de experiências em projetos individuais e evolução do conhecimento seja difícil. Além disso, destaca mais principalmente a reutilização de produtos GQM que lições aprendidas considerando a aplicação do enfoque GQM.

7 Avaliação de Tecnologias para Reutilização de Experiências em Engenharia de Software

Este capítulo analisa diferentes enfoques no campo da reutilização de conhecimento e como eles contribuem para os requisitos apresentados na Seção 4. Já que o objetivo deste trabalho é fornecer uma plataforma de suporte integrado para a reutilização de experiências em mensuração na prática, é dado enfoque tanto ao estado da prática quanto ao estado da arte dessas tecnologias. Exemplos dessas abordagens disponíveis em ambientes industriais ou diretamente aplicáveis na prática são Raciocínio Baseado em Casos, Recuperação de Informação, Hipertexto e Sistemas de Gerência de Bancos de Dados. As seguintes avaliações são baseadas em [ABG+98, GAB+98, ABH+97, TA97a, GB97a, Alt96].

7.1 Raciocínio Baseado em Casos

Raciocínio Baseado em Casos (RBC) [AP94] é uma abordagem para a solução de problemas e aprendizado sustentado e incremental (ver Seção 8). A abordagem reutiliza primeiramente problemas e situações experimentadas, juntamente com suas soluções para a resolução de novos problemas, apoiando a representação de qualquer tipo de artefatos por meio de um esquema formal variado (REQ2). Casos podem incluir vários tipos de conhecimento, por exemplo, textos, taxonomias, listas. A referência a outros casos permite a modelagem de interdependências entre artefatos. Com o objetivo de apoiar projetos de software, experiências de projetos similares antigos podem ser resgatados e as soluções correspondentes podem ser reutilizadas/adaptadas à situação atual. Caracterizações do contexto do qual as experiências tiveram origem são explicitamente capturadas como parte da descrição do problema, mesmo que não sejam fornecidos ainda suportes metodológicos para a determinação de características importantes em um domínio de Engenharia de Software em particular (REQ3). Generalizações de diferentes níveis de abstração de casos podem ser representados, bem como, domínio geral de conhecimento. Para a recuperação de adequados candidatos a reutilização, CBR oferece um modo de resgatar um conjunto de candidatos com atributos semelhantes baseados na fonte de informações disponível lidando também com

informações incompletas e inconsistentes (REQ4). Uma das maiores vantagens de CBR é seu aprendizado sustentado e incremental inerente que ocorre como um produto natural da solução do problema pela revisão e captura de experiências a cada vez que um novo problema for resolvido (REQ5). Um sistema baseado em casos pode ser inicialmente desenvolvido baseado somente em um pequeno conjunto de casos, e aumentado continuamente com novas experiências acumuladas. A abordagem suporta explicitamente aprendizado por exemplos, fornecendo meios para representar formalmente experiências consistindo de pares problema/ solução bem como informações adicionais. Considerando a manutenção da base de experiências, há pouco suporte fornecido pela modificação de esquemas de caracterização, por exemplo, para a adição e retirada de atributos. A adaptação, formalização e generalização de artefatos não são diretamente sustentados, bem como a re-estruturação automatizada das relações semânticas entre casos. Desse modo, o uso de relações semânticas em grandes bases de experiências é limitado.

CBR foi reconhecido como uma abordagem promissora para a operacionalização da organização do aprendizado no domínio da Engenharia de Software [GRW+99,GT99,ABG+98,ABT98,Gre98,AW97,TA97b,KS96]. Aplicações são desenvolvidas em diferentes áreas, como captura das melhores práticas [ANT99,BSA+97, FMV97,Hen97,ABS96,SM95,Hen92], predição de esforço [FWD97,MVP92], aquisição de requisitos [HC97,MR96,MS92,Mai91], sistemas de reutilização de código [GF97,FGG+96,Hen95,OHP+92], projeto de sistema de informação [Cza97,KL97], gestão de qualidade [LHI97] ou gerenciamento de mudanças [LS98].

A efetividade de recuperação desses sistemas varia da simples procura de capacidades até métodos de recuperação baseados em experiências. Entretanto, a maioria desses sistemas não considera explicitamente diferentes propósitos de reutilização, não oferece métodos de recuperação flexíveis e parametrizáveis e medidas de similaridade adaptáveis as metas específicas de reutilização (REQ1).

7.2 Recuperação de Informação

Recuperação de Informação (RI) [SM83] representa uma abordagem onde informações em forma de documentos de texto não estruturados (REQ2) podem ser acessados através de palavras-chave. Essas palavras expressam o principal assunto relativo a informação solicitada. Partes do conhecimento são recuperadas baseadas na especificação de um conjunto de palavras-chave (REQ4). Nos primeiros sistemas RI, a requisição era formulada como uma combinação booleana de palavras-chave. A maioria dos sistemas RI mudaram e utilizam um modelo de espaço vetorial no qual cada lista de palavras-chave é tratada como um vetor em um espaço n-dimensional. O modelo de espaço vetorial é mais flexível que o modelo booleano, já que os documentos podem ser ordenados por sua distância à requisição e o mais próximo pode ser relatado primeiro. Artefatos similares podem ser identificados pela procura na Base de Experiências ou incluindo relações específicas no processo de recuperação, por exemplo, afinidade léxica ou tesauri. Entretanto, em grande Bases de Experiências, a recuperação é restrita a uma seleção de palavras-chave bem feita. Características de contexto podem ser associadas aos documentos somente em forma de palavras-chave (REQ3). RI inclui técnicas para indexação automática de documentos texto resultando em uma descrição curta de cada documento, permitindo também recuperação através de índices Novos documentos podem ser adicionados na base de dados e indagados automaticamente. Como o conceito de RI apresentado não considera qualquer estrutura entre os documentos, a estrutura de índice pode ser automaticamente refeita em caso de modificação ou retirada de artefatos.

Devido ao explosivo crescimento da Internet, sistemas de recuperação de informação em forma de procura se tornaram populares. Codificação e procura em simples texto têm sido usados em um grande número de bibliotecas de software no domínio da engenharia de software (Por exemplo, [MBK91,FN87,BAB+87,FS84]).

Entretanto, existem algumas desvantagens do enfoque da recuperação de informação considerando as necessidades de reutilização de experiências em mensuração. Como o objetivo da RI é a recuperação de artefatos em bases de dados, o aprendizado crescente e sustentado não é freqüentemente considerado (REQ5). A adaptação, formalização e generalização dos artefatos não são suportados. E como o foco principal da recuperação de

informação é manipular grandes quantidades de dados para uma grande gama de tarefas, uma indexação e mecanismo de recuperação genéricos são utilizados (REQ1). Assim, somente eficiência e precisão reduzidas na recuperação podem ser alcançadas.

7.3 Abordagem Hipertexto

Um hipertexto é uma estrutura que permite ao usuário se mover de um lugar ao outro em um documento de texto através de *links* (REQ2). Os blocos básicos de construção em um hipertexto são nós e *links*. Cada nó, que pode ser de diferentes tipos, é associado à uma unidade de informação. Tipos de nós dependem de vários critérios, por exemplo, classe de dados armazenados ou domínio do objeto representado. *Links* definem relacionamentos entre nós fonte e destino. *Links* são acessados a partir do nó fonte e podem ser utilizados para acessar o nó destino. Atuais sistemas de hipertexto fornecem aos usuários ferramentas de interfaces de usuários que permitem inspecionar o conteúdo de nós e navegação flexível através de uma rede de nós. Além de permitir aos usuários cruzar *links* de acordo com suas vontades, sistemas de hipertexto fornecem aos usuários caminhos pré-definidos através da rede e a habilidade de especificar condições de procura para a seleção de nós. Sua procura pode ser baseada em conteúdo ou estrutural, dependendo da topografia da rede hipertexto. Ainda assim, não há suporte para recuperação baseada em similaridade (REQ4) ou adaptação de mecanismos de recuperação a várias metas (REQ1). Em geral, a abordagem hipertexto não suporta manutenção incluindo a contínua atualização da base de conhecimento e criação do domínio geral de conhecimento, embora o usuário possa manualmente criar, modificar ou retirar padrões e hiperlinks (REQ5). No entanto, para o suporte efetivo na aquisição de novas experiências, os mecanismos existentes devem ser refinados em relação a uma tarefa específica com o objetivo de fornecer um auxílio mais explícito. Hipertextos existentes e seus *links* podem ser completamente revisados pelo usuário, embora, em geral, nenhum suporte específico esteja disponível. Então são restritos a uma pequena base de experiências. Para cada documento, o contexto, do qual teve origem pode ser capturado, apesar da determinação de características relevantes não ser assistida (REQ3). A força do hipertexto é a utilização da tecnologia WWW, que permite a representação de tipos arbitrários de relações usando hiperlinks, recuperação via hiperlinks e navegação. Exemplos de sistemas baseados em

conhecimento utilizando a abordagem hipertexto no domínio da Engenharia de Software são [HK97,Tru97,IK96,Nie90,LJ88].

7.4 Sistemas de Gerenciamento de Bases de Dados

Um outro enfoque são os Sistemas de Gerência de Bases de Dados (SGBD)[1] que são amplamente usados na prática para o armazenamento estruturado e recuperação baseada em consulta em uma grande quantidade de informação. Em SGBDs, tipos diferentes de artefatos podem ser modelados através da definição de tabelas com seus atributos em um esquema de base de dados (REQ2). Experiências podem ser capturadas em uma base de dados enquanto representam o problema por meio de uma tabela no esquema da base de dados, que referência a solução dada pelo respectivo artefato. Características de contexto podem ser representadas como atributos nas tabelas da base de dados. Entretanto, sua aquisição e utilização para recuperação não é assistida (REQ3). Relacionamentos entre objetos são modelados através da definição explícita dos atributos que gerenciam o relacionamento. A recuperação dos artefatos é facilitada através de pesquisa, recuperação baseada em navegação e recuperação baseada em consulta, enquanto nem todos os atributos requeridos tem que ser necessariamente especificados (REQ4). Como a recuperação em SGBD é baseada na identificação exata, a recuperação artefatos de identificação parcial ou "similares" não é suportada (REQ4). Diferentes pedidos de recuperação podem ser formulados em relação ao objeto recuperado. Mesmo assim, a adaptação do método de recuperação em si, é limitado (REQ1). Novos artefatos podem ser incluídos no SGBD quando se tornam disponíveis (REQ5). Suporte para adição ou retirada de atributos é fornecida, bem como a atualização da semântica das relações pelas regras de integridade. Entretanto, a adaptação, formalização e generalização de qualquer artefato não são assistidas.

Exemplos de Sistemas de Gerência de Bases de Dados usados no domínio da Engenharia de Software são [TZ97,BCM+92,SME92,BR88].

1. Aqui nós enfocamos em sistemas de banco de dados tradicionais.

7.5 Discussão

Com base na avaliação da reutilização de conhecimento tecnologias relacionadas a importantes necessidades para o desenvolvimento de uma plataforma integrada de suporte visando a reutilização de experiências em mensuração na prática, CBR é a abordagem mais promissora com o objetivo de operacionalizar a reutilização de experiências em mensuração em ambientes industriais (ver Tabela 5) [GAB+98, ABG+98,Alt97].

Requisitos	Raciocínio Baseado em Casos	Recuperação de Informação	Abordagem Hipertexto	SGBD
REQ1. Recuperação orientada a metas	o	-	-	o
REQ2. Representação compreensiva de experiências em mensuração de software	+	o	o	o
REQ3. Caracterização e organização de experiências	o	o	-	o
REQ4. Recuperação baseada em similaridade de lições aprendidas relevantes baseada em informações incompletas e inconsistentes	+	-/o	-	-
REQ5. Aprendizado contínuo e incremental a partir de projetos individuais	+	-	-	-
				+ suporte alto; o suporte médio; - suporte baixo

Tabela 5 Comparação das tecnologias

A maior vantagem de RBC nesse contexto é a sua capacidade de realizar recuperações baseadas em similaridade para todos os tipos de artefatos e seu contínuo e incremental aprendizado como parte integrada do processo de reutilização. Além de seu foco principal em experiências, uma vantagem particular de RBC é que pode ser estendido usando representações adicionais de conhecimento a capacidades de aprendizado (por exemplo, aprendizado indutivo de árvores de decisão [Alt96]). Entretanto, RBC ainda tem várias restrições, e assim um suporte ótimo pode ser somente alcançado pela melhoria da abordagem e pela adaptação de sua aplicação no domínio da Engenharia de Software. Suporte apropriado, especialmente para a recuperação de experiências orientada a metas, é necessário para que se possa fornecer suporte compreensivo para várias metas de reutilização. Além disso, é

necessário uma orientação detalhada para a elaboração de índices relevantes com o objetivo de descrever o contexto no qual as experiências são válidas.

8 Raciocínio Baseado em Casos

O Raciocínio Baseado em Casos (RBC) [Wes95,AP94,Kol93] é um paradigma para a resolução de problemas que em muitos aspectos é fundamentalmente diferente de outras atividades da inteligência artificial. Ao invés de criar associações sobre relacionamentos generalizados entre descrições de problemas e conclusões, o RBC tem capacidade para utilizar o conhecimento específico de experiências antigas e situações de problemas concretos (casos). Um novo problema é resolvido pela busca de um caso similar no passado e sua reutilização no novo problema. Uma segunda diferença importante é que RBC é também uma abordagem para a aprendizagem sustentada e incremental, uma vez que uma nova experiência é retida a cada vez que um problema é resolvido, tornando-se imediatamente disponível para problemas futuros.

Raciocínio pelo reuso de casos passados é uma forma poderosa frequentemente utilizada por seres humanos para resolver problemas. Pessoas usam casos passados como modelos quando estão aprendendo a resolver problemas, particularmente em aprendizados recentes.

Deve-se notar que caso usualmente denota uma situação de problema. Uma situação anteriormente experimentada, a qual é capturada e aprendida em uma forma que pode ser reutilizada na resolução de futuros problemas, é referenciada como um caso passado, caso armazenado ou caso retido. Correspondentemente, um novo caso ou caso não resolvido é a descrição de um novo problema a ser resolvido. Um caso típico é usualmente assumido por conter um certo grau de riqueza de informações contidas em si e uma certa complexidade com respeito a sua organização interna. Possui também outras características como: a possibilidade de ser modificado ou adaptado, e a solução recuperada pode ser aplicada em diferentes contextos de solução de problemas.

8.1 A história do Raciocínio Baseado em Casos

As raizes do raciocínio baseado em casos na inteligência artificial são fundamentadas nos trabalhos de Schank e Abelson em Memória Dinâmica e na função central de lembrança

de situações passadas (casos, episódios) e padrões de situações que tem na solução de problemas e aprendizado [SA77]. Outras trilhas no campo de CBR vieram do estudo do raciocínio analógico e, mais tarde, das teorias de formação de conceitos, resolução de problemas e aprendizagem experimental dentro da filosofia e psicologia.

O primeiro sistema que pode ser chamado de um raciocinador baseado em casos foi o sistema CYRUS [Kol93], um sistema de perguntas e respostas com o conhecimento de várias viagens e reuniões do secretário de estado dos Estados Unidos Cyrus Vance. O CYRUS é baseado no modelo de memória dinâmica e na teoria dos pacotes de organização de memória (MOP) na resolução de problemas e aprendizagem.

Outras bases para RBC foram desenvolvidas por Porter e seu grupo na Universidade de Austin, no Texas. Eles inicialmente endereçaram o problema da maquina de aprendizagem do conceito de aprendizagem para tarefas de classificação. Isto liderou o desenvolvimento do sistema PROTOS [Bar88], o qual destacava a integração geral do conhecimento do domínio e conhecimento específico de casos em uma estrutura de representação unificada.

Uma outra contribuição para RBC foi o trabalho de Rissland e seu grupo na Universidade de Amherst, Massachusetts. Com vários cientistas da lei no grupo, eles estavam interessados no papel do raciocínio na precedência em julgamentos legais. Casos (precedencias) aqui não são usadas para produzir apenas uma simples resposta, mas para interpretar uma situação na corte, e para produzir e avaliar argumentos para as duas partes. Isto resultou no sistema HYPO, e mais tarde na combinação de baseado em casos e baseado em regras, no sistema CABARET. Phyllips Koton do MIT estudou o uso de raciocínio baseado em casos para optimizar a performance de um sistema baseado em conhecimento, onde o domínio (falhas do coração) são descritas por profundidade e modelos causais. Isto resultou no sistema CASEY, no qual baseado em casos e baseado em modelo de profundidade foram combinados.

Na Europa, a pesquisa em RBC foi iniciada um pouco depois que no Estados Unidos. Entre os últimos resultados estava o trabalho em RBC para diagnóstico técnico complexo com o sistema MOLTKE, feito por o grupo do Richter na Universidade de Kaiserslautern. Isto levou ao sistema PATDEX [Wes95]e mais tarde com diversos outros sistemas e métodos.

Atualmente, as atividades de RBC estão dissiminado-se, e rapidamente aumentando o número de papéis em RBC em quase toda publicação em inteligência artificial.

8.2 Fundamentos do Raciocínio Baseado em Casos

As tarefas centrais de todos os métodos de raciocínio baseado em casos tem em comum a identificação de uma situação de problema atual, a procura por um caso no passado que seja similar ao novo caso, a utilização deste caso para sugerir uma solução para o problema corrente, a avaliação da solução proposta e a atualizaçao do sistema pela aprendizagem desta nova experiência [AP94].

8.2.1. Representação de casos

Um caso é uma peça contextualizada do conhecimento representando uma experiência. Ele contém uma lição passada, isto é o conteúdo do caso e o contexto no qual a lição pode ser usada. Um caso pode ser o relato de um evento, uma estória, ou algum registro típico.

O problema descreve o estado do mundo quando o caso ocorreu, e solução demonstra a solução derivada para um determinado problema. Uma outra maneira de visualizar isto é em termos de espaço do problema e espaço da solução. A descrição do novo problema a ser resolvido é posicionada no espaço do problema, a recuperação identifica o caso com a mais similar descrição do problema e a solução armazenada é encontrada. Se necessário, ocorrem adaptações e uma nova solução é criada. Este modelo conceitual de RBC assume que há um mapeamento direto de um-para-um entre os espaços de problema e solução.

A representação do problema em RBC é primariamente o problema de decidir o que será armazenado no caso, procurando a estrutura apropriada para descrever o conteúdo de um caso, e decidir como a memória do caso pode ser organizada e indexada para a recuperação efetiva e reuso. Um problema adicional é como integrar a estrutura de memória do caso no modelo do conhecimento geral do domínio, para estender o conhecimento incorporado.

8.2.1.1. O modelo de memória dinâmica

O primeiro sistema que pode ser referenciado como um sistema de raciocínio baseado em casos foi o CYRUS, baseado no modelo de memória dinâmica [SA77]. A memória do caso neste modelo é uma estrutura hierárquica a qual é chamada pacotes de organização de memória episódica (E-MOP's) também referenciado como episódios generalizados (GE). A idéia básica é organizar casos específicos os quais compartilham propriedades similares sob uma estrutura mais geral (um episódio generalizado). Um episódio generalizado contém três diferentes tipos de objetos: normas, casos e índices. Normas são qualidades em comum a todos os casos sob o GE. Índices são qualidades as quais discriminam os casos uns dos outros. Um índice pode apontar para um episódio generalizado mais específico ou diretamente para um caso. Um índice é composto por dois termos: O nome do índice, e o valor do índice. A memória de casos é inteiramente uma rede descriminatória onde um nó é ou um episódio generalizado, o nome de um índice, o valor de um índice ou um caso.

8.2.1.2. O modelo categoria & exemplar

O sistema PROTOS [Bar88] propõe uma forma alternativa para organizar casos em uma memória de casos. Casos são também referenciados como exemplares. A base psicológica e filosófica para este método é a visão de que no mundo real, conceitos naturais podem ser definidos extensionalmente. Além disso, diferentes características são assinaladas com importâncias diferentes na descrição de um caso associado a uma categoria. A memória do caso é embutida em uma estrutura de rede de categorias, relações semânticas, casos e ponteiros de índices. Cada caso é associado com uma categoria. Um índice pode apontar para um caso ou uma categoria. Os índices são de três tipos: ligações para características apontando de descritores do problema (características) para casos ou categorias (lembranças), ligações de casos apontando de categorias para os casos associados (ligação de exemplares), e ligações de diferenças apontando de casos para os casos vizinhos que apenas diferem em um ou um número pequeno de características. Uma característica é, geralmente, descrita por um nome e um valor.

Entre esta organização de memória, as categorias são interligadas em uma rede semântica, a qual também contém as características e estados intermediários encaminhados

por outros termos. Esta rede representa uma formação do conhecimento geral do domínio o que habilita o suporte explanatório de algumas tarefas de RBC.

8.2.2. Indexação

Muitos sistemas de bancos de dados utilizam índices para acelerar a recuperação de dados. Um índice é uma estrutura computacional de dados que pode ser colocada na memória e rapidamente pesquisada. Isto significa que não é necessário percorrer cada registro armazenado no disco, o que pode ser lento. RBC também utiliza índices para acelerar a recuperação. Informações em um caso podem ser de dois tipos: Informações indexadas que serão utilizadas para recuperação e informações não indexadas que provêm informação contextual de valores para o usuário, mas não são usadas diretamente na recuperação. Índices devem ser preditivos, endereçar os propósitos para os quais o caso será usado, ser abstrato para permitir ampliação futura da base de casos, e ser concreto para ser reconhecido no futuro [Kol93]. Existem vários métodos de indexação automática que baseado num grande conjunto de casos automaticamente determinam os índices relevantes, tais como indexação por características e dimensões que tendem a ser preditivas por todo o domínio, indexação baseada na diferença, a qual seleciona índices que diferenciam um caso de outros, métodos de generalização baseados em similaridade e explanação, que produzem um apropriado conjunto de índices para casos abstratos a partir de casos que compartilham um conjunto comum de características, ou métodos indutivos de aprendizagem, que identificam características preditivas que são então usadas como índices, para aplicações práticas, índices podem ser escolhidos automaticamente, manualmente, ou pelas duas técnicas [Kol93].

8.2.3. Armazenamento

O armazenamento de casos é um aspecto importante no desenho da eficiente de sistemas RBC, no qual é refletido a visão conceitual do que é representado no caso e dos índices que caracterizam o caso. Um sistema baseado em casos pode ser organizado em uma estrutura gerenciável que suporta a busca eficiente e métodos de recuperação. Um equilíbrio é encontrado entre os métodos de armazenamento que preservam a riqueza dos casos e seus índices e métodos que simplificam o acesso e a recuperação dos casos relevantes.

8.2.4. Recuperação

A recuperação dos casos é dependente do método de representação utilizado. Em geral duas técnicas são usadas por aplicações de RBC comerciais:

- a recuperação pelo vizinho mais próximo (*nearest-neighbor retrieval*)
- e recuperação indutiva (*inductive retrieval*).

8.2.4.1. Recuperação pelo Vizinho Mais Próximo

A recuperação pelo vizinho mais próximo é conceitualmente uma técnica simples [Wat97]. O que é preciso é calcular a distância relativa entre um caso alvo e os outros casos. Aquele que obtiver o menor valor é o caso vizinho mais próximo. Esta solução utiliza-se de uma medida de similaridade [Ric95], que pode utilizar-se de pesos diferentes para cada atributo, onde a soma da similaridade de todos os atributos é calculada. Algorítmos similares a estes são usados em muitas ferramentas de RBC para realizar esta recuperação.

8.2.4.2. Recuperação Indutiva

Esta técnica utilizada por muitas ferramentas RBC envolvendo um processo chamado indução [Alt96]. Indução é uma técnica desenvolvida por pesquisadores de aprendizado de máquina para extrair regras ou construir árvores de decisão a partir de dados passados. Em sistemas RBC, o baseamento em casos é analisado por um algorítmo de indução para produzir uma árvore de decisão que classifica ou indexa os casos. O algorítmo de indução mais largamente utilizado é chamado de ID3. Este algorítmo constrói uma árvore de decisão a partir de uma base de casos. Utiliza-se de uma heurística chamada ganho de informação para procurar o atributo mais promissor, a partir do qual a árvore será dividida ao meio.

Estas duas técnicas são amplamente usadas em ferramentas de RBC, para decidir qual delas é melhor pode requerer experimentação e certamente requer experiência.

8.2.5. Adaptação

Quando um caso é recuperado, o sistema RBC estará atento para reutilizar a solução sugerida pelo caso recuperado. Em muitas circunstâncias a solução pode ser suficiente. Porém,

em outras circunstâncias a solução pode necessitar de uma adaptação da solução recuperada para as necessidades do caso atual. Adaptações procuram por diferenças proeminentes entre o caso recuperado e o caso corrente, e então aplicar formulas e regras que tomam aquelas diferenças no relatório sugerindo a solução final. Em geral existem dois tipos de adaptação em RBC [Kol93]: Adaptações estruturais que aplicam regras e formulas de adaptação diretamente na solução armazenada nos casos, ou adaptações derivativas, que reutilizam regras ou formulas que geraram a solução original para produzir uma nova solução para o problema corrente. Neste método a seqüência de planejamento que construiu a solução original deve ser armazenada como um atributo adicional ao caso. Esta adaptação pode ser apenas utilizada para domínios que estão bem entendidos.

8.2.6. Aprendizado

Este é o processo de incorporação que é útil para reter a partir do novo episódio de um problema resolvido na base de conhecimento existente. A aprendizagem com sucesso ou não da solução proposta é disparada pela saída de uma avaliação e possível reparo. Isto envolve a seleção de qual informação do caso é retida, em que formato é retida, como indexar o caso para posterior recuperação para problemas similares, e como integrar o novo caso na estrutura de memória [AP94].

A etapa de avaliação pode ser programada para execução automática ou com participação do usuário. A avaliação tem por objetivo avaliar a qualidade da solução adaptada ao problema de entrada para definir se esta tem condições de ser adicionada a memória.

8.2.6.1. Extração

Em RBC, a base de casos é atualizada não interessando a forma pela qual o problema foi resolvido. Se ele utiliza um caso prévio, um novo caso pode ser construído ou um caso antigo pode ser generalizado. Se o problema foi resolvido por outros métodos, incluindo perguntar ao usuário, um caso inteiramente novo será construído. Em qualquer caso a decisão que deve ser tomada sobre o que deve ser usado como fonte para o aprendizado. Falhas podem ser extraídas ou retidas como casos de falha separados ou como casos completos de problema. Quando uma falha é encontrada, o sistema pode lembrar-se de uma falha similar prévia e utilizar o caso de

falha para melhorar o entendimento e corrigir a presente falha. A inclusão do caso adaptado, reutilizado e avaliado consiste na aprendizagem em um sistema RBC.

8.2.6.2. Indexação

O problema da indexação é o problema central e muitas vezes focado problema em RBC. Ele soma as decisões sobre qual tipo de índices a usar para futura recuperação, e qual estrutura para procurar espaço para índices. Isto é atualmente o problema da aquisição de conhecimento, e precisa ser analisado a parte da análise e modelagem do domínio do conhecimento.

8.2.6.3. Integração

Este passo final da atualização da base de conhecimento com um novo caso de conhecimento. Se nenhum caso e índice será construído, é o passo principal da retenção. Pela modificação dos índices dos casos existentes, Sistemas RBC aprendem a tornar a avaliação mais similar. O ajuste dos índices existentes é uma importante parte da aprendizagem do RBC. Neste caminho, a estrutura de índices tem o papel de ajustar e adaptar a memória de casos para esse uso. O caso recém aprendido pode ser finalmente testado, re-entrando o problema inicial e vendo o que o sistema recupera.

8.3 Aplicações do Raciocínio Baseado em Casos

A princípio qualquer problema pode ser implementado em RBC. Não é necessário que exista um algorítmo para a resolução de um problema para resolvê-lo em RBC. Inclusive aponta-se para uma de suas vantagens, a capacidade de prover a resolução de um problema sem a necessidade de algorítmos ou a aquisição de conhecimento. Basta que o mesmo problema (ou outro similar) já tenha sido resolvido. O afunilamento da resposta com relação à adequação de determinado problema, dar-se-á na medida que avaliemos técnicas competitivas e a disponibilidade de dados. Quanto a disponibilidade de dados, a existência de um banco de dados contendo um número suficiente de casos representa um avanço no desenvolvimento de um RBC. Os dados de um banco podem ser vistos como casos.

As aplicações do RBC, podem ser classificadas em dois tipos de problemas principais: tarefas de classificação e tarefas de sintetização. Tarefas de classificação cobrem uma grande faixa de aplicações que compartilham de características comuns, onde um novo caso é identificado com os que estão na base de casos de forma a determinar o tipo ou classe do caso em si. Podem ser tarefas como: diagnóstico, previsão, avaliação, controle de processos ou planejamento. Tarefas de sintetização tentam criar uma nova solução a partir da combinação de partes de soluções préviamente aplicadas, podendo ser tarefas de desenho, planejamento e configuração [Wat97,Alt97].

8.3.1. Tarefas de classificação

Tarefas de classificação são muito comuns nos negócios e no dia-a-dia. Podem ser reconhecidos pela necessidade de se encontrar um objeto ou evento entre outros em conjunto a partir da qual uma resposta pode ser inferida. Muitos destes objetos ou eventos que compõe o conjunto de casos podem ser classificados em grupos que são previsíveis por sua saída. A saída é geralmente um atributo de cada e é comummente como tentamos classificar os casos. Assim, é perfeitamente possível classificar casos de diferentes formas.

Tarefas de classificação são facilmente implementadas porque adequam-se bem ao ciclo do RBC, os casos tendem a ser fáceis de representar e fáceis de coletar e, os algorítmos de recuperação usados na maioria das ferramentas de RBC são classificadores.

8.3.2. Tarefas de sintetização

Tarefas que requerem sintetização são comuns em ambientes comerciais e industriais, mas são difíceis de implementar. Isto acontece porque em geral é fácil encontrar um artefato em um conjunto de artefatos protótipados do que construir um artefato a partir de uma especificação tarefas de classificação requerem o reconhecimento de características enquanto tarefas de sintetização requerem colocar a característica correta, no lugar correto e na ordem correta.

Sistemas de sintetização baseada em casos trabalham no domínio de desenho ou planejamento, geralmente tendendo a tentar simplificar o processo criativo através da

produção de conhecimento para um bom desenho ou planejamento a partir do qual o planejamento final pode ser construído.

8.4 Conclusões

Pode-se dizer que o Raciocínio Baseado em Casos, permitem-nos a resolução dos mais variados tipos de problemas, bem como aprendizado de máquina, e uso do conhecimento específico de um domínio, sem, no entanto, a necessidade da elaboração de complexos algorítmos que resolvam o problema, bastando apenas que o mesmo já tenha sido resolvido antes, e sua solução armazenada em uma base de casos, que posteriormente serão utilizados.

Problemas resolvidos a partir do reutilização de casos aprendidos no passado, passam a ser parte das possíveis soluções de outros problemas futuros, constituindo-se assim um mecanismo de aprendizagem contínua, tornando o sistema a cada caso, mais especializado do que anteriormente, além do que o campo de aplicação do raciocínio baseado em casos é práticamente infinito, podendo ser aplicado a quase todos os tipos de problemas existentes.

O espaço da solução é reduzido, uma vez que é possível recuperar uma informação rápidamente, ao contrário de outras aplicações que precisam buscar a solução em uma vasta base de conhecimento, uma vez que o problema deve ser identificado apenas o suficiente para identificar o problema, sem no entanto haver a necessidade da compreensão total do problema, para apresentar uma solução.

9 Uma Abordagem Baseada em Casos para Reutilização de Lições Aprendidas em Mensuração de Software

Nesse capítulo, será apresentada uma metodologia para a operacionalização de Fábrica de Experiência em estudo dando suporte a aspectos tecnológicos relacionados a aquisição sistemática e comunicação de lições aprendidas em mensuração através da empresa.

Considerando os requisitos específicos para uma plataforma de suporte integrado para reutilização de lições aprendidas em mensuração durante o planejamento de programas de mensuração, a metodologia é organizada através da abordagem da FE aplicando técnicas de raciocínio baseado em casos para a solução do problema e aprendizado sustentado progressivo. A metodologia é baseada no enfoque REMEX abordando a recuperação de experiências em Engenharia de Software [Gre99,GAB99,GWB98,GB97a] pela adaptação e melhoria de técnicas desenvolvidas para reutilização de lições aprendidas em mensuração.

Na *Base de Conhecimentos de Lições Aprendidas GQM* (GQM-LL-KB), experiências em tecnologia de mensuração em forma de casos contendo *Casos de Experiência Problema/ Solução GQM* (GQM-PSEC), expressando o problema ocorrido e a estratégia de solução adotada no passado. Durante o planejamento de novos programas de mensuração, casos apropriados com características similares são recuperados da GQM-LL-KB baseados na descrição da situação atual, por exemplo, organização ou tarefa de mensuração relacionada e fornecida ao usuário as candidatas a reutilização. O usuário pode explorar os casos recuperados, selecionar o mais apropriado e se necessário, adaptar o caso selecionado a necessidades específicas da situação atual. Já que a GQM-LL-KB é utilizada como meio de comunicação para compartilhar experiências na empresa, a recuperação e comunicação de casos é enfatizada, tanto quanto a sua adaptação automatizada a características da situação atual. Integrado ao processo de solução do problema está a aquisição de novas experiências. Cada vez que uma experiência passada é reutilizada com o objetivo de resolver um novo problema, experiência descrevendo a situação do problema atual e como foi resolvido é adquirida e integrada a GQM-LL-KB.

Nas seções seguintes, descreve-se a representação de conhecimento experimental em

mensuração , sua recuperação, aquisição e integração em detalhes.

9.1 Representação do conhecimento na GQM-LL-KB

Nesta seção, descreve-se dois principais tipos de representação de conhecimento na GQM-LL-KB: lições aprendidas em mensuração em forma de GQM-PSECs e conhecimento do domínio da área de mensuração.

9.1.1. Representação de conhecimento de experiências em mensuração

O objetivo da abordagem é capturar e compartilhar *know-how* sobre a aplicação, utilização e adaptação do planejamento de programas de mensuração de software seguindo o enfoque GQM. Assim, o conhecimento tácito teve que ser adquirido do pessoal de garantia de qualidade e explicitado, habilitando sua representação e facilitando sua comunicação. Entretanto, a codificação do conhecimento tácito é difícil. A solução é a representação desse conhecimento em forma de episódios, descrevendo mais narrativamente uma situação concreta que aconteceu no passado, do que derivando conhecimento generalizado e abstrato, por exemplo em forma de regras e heurísticas. A captura de lições aprendidas que explicitamente documentam quais problemas ocorreram no passado e como foram resolvidos, é uma forma efetiva de comunicar esse tipo de conhecimento. Dessa forma, como uma importante fonte para guiar a aplicação de mensuração na prática, conhecimento sobre experiências em mensuração em forma de lições aprendidas é armazenado na GQM-LL-KB [GWB98,GMA+98,GB97a]. A reutilização de lições aprendidas pode prevenir possíveis falhas e guiar a solução conforme o contexto da aplicação. Devido a natureza específica das experiências, que suportam a manipulação de exceções, este é capturado pela declaração de um problema ocorrido durante o planejamento de uma programa GQM e suas soluções experimentadas em projetos de software passados. Esse conhecimento é primeiramente representado em forma de casos concretos, denotados por GQM-PSEC, que são descrições específicas ao contexto de lições aprendidas particulares obtidas durante o planejamento de programas de mensuração passados. Cada caso é relacionado ao um episódio problema/ solução específico. Um GQM-PSEC basicamente inclui a descrição do problema, sua causa, a

solução adotada e informação sobre os resultados. Com o objetivo de facilitar a recuperação efetiva e fornecer guia detalhado para a aquisição de experiências GQM, essas partes básicas são refinadas em dimensões detalhadas e associadas pela descrição de contexto com o objetivo de permitir a identificação de experiências relevantes em uma situação particular. Informações básicas, como autor, data de criação, etc. são adicionadas objetivando a decisão de reutilização de lições aprendidas. Informações explícitas em reutilização de lições aprendidas são associadas ao caso, fornecendo guia adicional pela indicação de possível necessidade de adaptação ou problemas relacionados com a reutilização das respectivas experiências.

Comentários adicionais são possivelmente armazenados com o objetivo de garantir a representação compreensiva de experiências além das dimensões definidas. Em resumo, as principais dimensões de um GQM-PSEC são:

- **Descrição de Contexto.** O contexto organizacional e específico ao projeto do qual a experiência originou é descrito (por exemplo, nome da organização, linguagem de programação, domínio de aplicação). Com o objetivo de manter uma descrição mínima, somente características que são relevantes ao GQM-PSEC em questão são listados. Por exemplo, o tipo de software (por exemplo, software embutido) pode ser irrelevante para um problema considerando a validade da coleção de dados. Apesar de que a duração do projeto ou tamanho da equipe de projeto são importante, se eles causam impacto nas causas do problema.
- **Problema.** É descrito o problema ocorrido durante o processo de planejamento de mensuração.
- **Causa do problema.** A causa do problema é descrita explicitamente, se conhecida. O objetivo é prevenir a repetição de possíveis problemas em programas de mensuração futuros baseado no conhecimento explícito sobre as causas que originaram problemas.
- **Solução.** É descrita a solução aplicada para resolver o problema. Esta orienta lidando com novos problemas durante a reutilização de estratégias de solução passadas em futuros programas de mensuração. Uma justificativa, explicitamente fornecendo um argumento para sua seleção é dado, destacando as interdependências entre a causa, sua explicação e a solução aplicada.
- **Resultado.** O resultado de saída obtido pela aplicação da solução é descrito com o objetivo de antecipar o resultado esperado em futuras reutilizações desse caso. Além da

captura de casos resolvidos bem sucedidos, também casos descrevendo tentativas de resolver o problema são armazenados. Esses casos apontam soluções que podem possivelmente falhar, quando aplicadas para resolver o problema atual e podem ajudar a prevenir a repetição de falhas no futuro.

- **Informações Básicas.** Com o objetivo de apoiar uma utilização apropriada de casos disponíveis e uma seleção rápida de informações relevantes, informações básicas em cada caso são fornecidas.
- **Reutilização de Informações.** Com o objetivo de guiar a decisão de reutilizar um caso e, se necessário, sua adaptação a situação atual, informações sobre reutilizações prévias em um caso específico são apresentadas.

Os GQM-PSECs são modelados pela utilização de uma representação formal, flexível, orientada a objetos, em forma de pares atributos/valores, baseada em [TG99,MBC+94] e são estruturadas em detalhe como mostra a Tabela 6 [GMA+98].

Atributo	**Descrição**
Caracterização de contexto	
Características	Todas as características relevantes ao contexto são listadas em forma de pares atributo-valor, denotados como tuplas: (característica, valor).
Problema	
Descrição do problema	O problema é descrito em forma de texto.
Objeto do problema	O objeto afetado pelo problema é declarado. O objeto afetado pode ser qualquer processo, produto ou modelo de recurso ou instância (ex.: coleção de dados, dados de tentativa, verificador).
Tarefa do problema	A tarefa no qual o problema ocorreu é declarada.
Função envolvida	A função organizacional envolvida no problema é declarada.
Meta não alcançada	A meta de uma determinada tarefa GQM que não foi realizada por causa do problema é declarada.
Comentário	Qualquer informação adicional ou comentário sobre o problema são declarados.
Causa(s) do problema(s)	

Tabela 6 Definição do GQM-PSEC

Descrição da causa	A causa do problema, se conhecida , é descrita em forma textual.
Objeto da causa	O objeto que causou o problema pode ser declarado. O objeto pode ser qualquer processo, produto, modelo de recurso ou instância.
Tarefa da causa	É identificada a tarefa que causou o problema, que pode ser diferente da tarefa de ocorrência do problema.
Função envolvida	Função da organização envolvida na causa do problema é declarada.
Obstáculos	Obstáculos em relação ao projeto de software que influenciara o problema, ex. em relação a disponibilidade de recursos, esforços ou duração, são declarados.
Comentários	Qualquer informação adicional ou comentário sobre a causa são adicionados como texto.
Solução	
Descrição da solução	A solução é descrita em forma textual.
Justificativa	A solução é justificada, enfocando interdependências entre a causa, sua explicação e a solução aplicada. A justificativa permite a avaliação adequação de uma solução passada em uma situação atual, enquanto que fornece um argumento explícito para sua seleção.
Comentários	Qualquer informação adicional ou comentários sobre a solução são declarados.
Resultado	
Avaliação de Resultados	A avaliação expressa explicitamente se o problema foi resolvido com sucesso pela aplicação da solução ou fracassou.
Descrição do resultado	Os resultados da solução aplicada são descritos em forma textual. Se a solução aplicada falhou em resolver o problema, uma explicação é dada sobre o porquê da meta da respectiva tarefa ainda não foi alcançada.
Comentários	São descritos informações adicionais ou comentários sobre os resultados.
Informações básicas	
Ponto de vista	É declarada a função de onde o conhecimento foi adquirido.
Representatividade	A representatividade do caso é dada em termos do número de projetos de software individuais a partir do qual foi derivado. Por exemplo, uma vez capturado de um projeto de software um caso pode ser confirmado em outros projetos, quando é reutilizado aumenta sua representatividade.
Data de criação	É declarada a data de criação do caso, quando foi integrado na GQM-LL-KB.
Posse	É declarado o autor do caso.
Direito de acesso	São definidos os direitos de acesso ao caso.
Reutilização de informação	

Tabela 6 Definição do GQM-PSEC

Pré-condições necessárias	São declaradas pré-condições para a reutilização da entidade descrita.
Adaptações esperadas	São descritas adaptações feitas no passado durante a reutilização da entidade e fatores relevantes que motivaram as adaptações.
Datas da reutilização	Datas da reutilização com o objetivo de fornecer uma visão geral de quando e quão frequente a entidade foi reutilizada são listadas.
Diretrizes para reutilização	É declarado um guia de como reutilizar a entidade.

Tabela 6 Definição do GQM-PSEC

Um exemplo de um GQM-PSEC é mostrado na Tabela 7.

Caracterização de contexto	organização	IntelliCar
	setor de aplicação	automóvel
	tamanho da equipe	10 desenvolvedores
	nível deexperiência	baixo

Tabela 7 Exemplo de um GQM-PSEC

	Problema	
	Descrição do problema	O entrevistado não coopera com o entrevistador
	Tipo de objeto do problema	execução da entrevista
	Tarefa do problema	GQM2.2.2 Aquisição de conhecimento
	Função do problema	entrevistado
	Meta não atingida	adquirir toda a informação relevante a meta
	Causa	
	Descrição da causa	falta de motivação para mensuração
	Objeto da causa	Treinamento dos participantes
	Tarefa da causa	GQM1 Estudo prévio
	Função(s) da causa	Engenheiro de garantia da qualidade
	Obstáculo(s)	Limitação de utilização de esforços em mensuração
	Solução	
	Descrição da solução	Explicar os objetivos do programa da mensuração e da entrevista e garantir a confiabilidade das informações coletadas no começo da cada entrevista
	Justificativa	Assim, o entrevistado pode entender os objetivos e eliminar alguma resistência em relação a mensuração
	Resultado	
	Descrição do resultado	Entrevistado participou ativamente da entrevista e compartilhou suas informações
	Avaliação do resultado	solução aplicada com sucesso
Informações Básicas	Ponto de vista	Engenheiro de garantia da qualidade
	Representatividade	2 episódios
	Data de criação	11.10.1999
	Posse	C. von Wangenheim
Informação de reutilização	Pré-condições necessárias	nenhum
	Adaptações esperadas	Dependendo do interesse e necessidade do entrevistado o nível e escopo da explicação pode variar.
	Datas da reutilização	01.01.2000
	Diretrizes para reutilização	nenhum

Tabela 7 Exemplo de um GQM-PSEC

9.1.2. Representação do conhecimento geral de domínio

Além da representação de exemplos explícitos de problemas e suas soluções de um projeto individual, a representação de conhecimento geral do domínio pode também facilitar a aquisição e reutilização de experiências [GWB98,GB98].

Nesse contexto, o conhecimento geral de domínio compreende a definição dos tipos de atributos da representação dos GQM-PSECs e um tesaurus em terminologia de mensuração. Definições de tipo [Gre99,TG99,GB98,TG98] modelam qualidades de objetos relacionados ao domínio de mensuração de software, tais como experiência do desenvolvedor ou categorizar conceitos, ex., linguagens de programação. Definições de tipo definem explicitamente o nome do tipo, seu supertipo, uma unidade (no caso de tipos numéricos), o possível intervalo de valores do tipo e medida de similaridades locais definido explicitamente a similaridade entre valores do respectivo tipo (ver Seção 9.1.2).

Por exemplo, o tipo do atributo "tarefa do problema" é definido como Símbolo Não Ordenado com os possíveis valores "Identificação da meta GQM", "Aquisição de conhecimento", "Desenvolvimento de questões GQM", etc. utilizando uma medida de similaridade local com o objetivo de determinar a similaridade entre os valores definidos.

Definições de tipo são utilizadas para classificar atributos. Eles suportam a aquisição consistente de experiências nos projetos, facilitam a avaliação da situação e também suportam o processo de adaptação manual de GQM-PSECs pela indicação explícita de alternativas. Tipos podem ser derivados a partir de um conjunto de tipos básicos pré-definidos (ver Figura 8):

- Booleano
- Número
- Texto (texto livre em linguagem natural)
- Data (em forma de dia/mês/ano)
- Símbolo (símbolos, que podem ser não-ordenados, ordenados ou representados em forma de taxonomia)

A partir do conjunto de tipos pré-definidos, tipos específicos de domínio podem ser

derivados como subclasses para adaptar a abordagem a um ambiente específico Baseadas na estrutura hierárquica das classes de tipo, as subclasses herdam as definições da superclasse se não as sobrescrevem no caso específico.

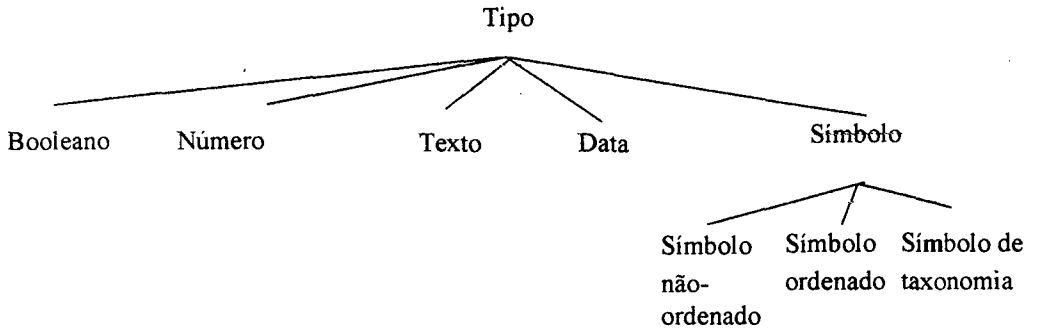


Figura 8 Tipo básicos

Exemplos são ilustrados na Tabela 8.

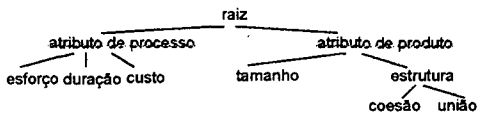
Nome do tipo	**Supertipo**	**Unidade (para tipos numéricos)**	**Intervalo de valores**	**Similaridade**
Pessoa-hora	Número	hora	[0,•]	Padrão
Escala	Símbolo ordenado	-	"nominal", "ordinal", "intervalo", "razão", "absoluto"	SimDeEscala
Atributo SW	Símbolo de taxonomia	-		Padrão

Tabela 8 Exemplos de definição de tipos

Como muitos atributos dos GQM-PSECs são do tipo texto, conhecimento geral de domínio adicional é representado na GQM-LL-KB em forma de um tesauro com o objetivo de apoiar a recuperação de GQM-PSEC adequado. Um tesauro é "um vocabulário controlado e dinâmico de descritores relacionados semântica e genericamente, que cobre de forma extensiva um ramo específico de conhecimento" [Aur96]. O tesauro lista explicitamente

sinônimos para termos de mensuração de software. Por exemplo, como sinônimos para o termo defeito, os termos erro ou falha podem ser usados.

Por representar explicitamente sinônimos na GQM-LL-KB a recuperação de casos similares é também apoiada mesmo para atributos de texto. Assim, valores similares em forma de sinônimos podem ser considerados e recuperados. Isto é especialmente importante, ainda que pareça ser simples e intuitivo a princípio, uma vez que a GQM-LL-KB que foi construída ao longo do tempo, baseada em casos adquiridos de diferentes pessoas, problemas irão surgir em uma percentagem baixa de documentos relevantes recuperados (*recall*) a menos que a variedade e consistência dos termos usados possa lidar sistematicamente com esse tipo de informação.

9.2 Recuperação e reutilização de lições aprendidas em mensuração

O planejamento de programas de mensuração GQM é apoiado pela reutilização de GQM-PSECs documentando lições aprendidas em mensuração a partir de programas de mensuração passados. Enquanto o engenheiro de garantia de qualidade está planejando o programa de mensuração baseado na abordagem GQM, ele/ela pode solicitar suporte baseado na reutilização de GQM-PSECs úteis para uma situação específica e a GQM-LL-KB é investigada e um conjunto de casos apropriados é recuperado da GQM-LL-KB como candidatos a reutilização. O usuário pode explorar os candidatos sugeridos a reutilização e selecionar o que melhor atende as necessidades atuais.

Com o objetivo de fornecer um suporte eficiente, vários requisitos têm que ser considerados no contexto da reutilização de lições aprendidas durante o planejamento de programas de mensuração (ver Seção 4).

Durante o planejamento de programas de mensuração GQM a GQM-LL-KB pode ser pesquisada para encontrar GQM-PSECs úteis para suportar várias tarefas de planejamento de mensuração (ex., aquisição de conhecimento ou desenvolvimento de medidas) para diferentes propósitos (ex., provenção de problemas, solução de problemas) em um ambiente específico. Visando fornecer suporte compreensivo para muitos objetivos, é utilizado um método de

recuperação orientado a metas [GAB00,GAB99,Gre99] que recupera um conjunto de casos de experiências relevantes relativos a uma meta específica de reutilização. Metas de reutilização são determinadas explicitamente especificando as seguintes dimensões:

Retornar	*<objeto>*
para	*<propósito>*
relativo a	*<processo>*
a partir de	*<ponto de vista>*
no contexto de	*<ambiente>*

Por exemplo, "retornar GQM-PSEC para guiar a solução do problema relativo a mensuração de software a partir do ponto de vista da equipe de garantia de qualidade no contexto da IntelliCar".

O método de recuperação orientado a metas habilita a parametrização flexível e dinâmica e adaptação do método de recuperação de acordo com a meta específica a ser obtida pela recuperação. Por exemplo, para a recuperação de uma estratégia de solução, índices relevantes podem ser a descrição do problema e a tarefa quando o problema ocorreu, apesar de que possíveis problemas com respeito a uma tarefa específica podem ser identificados baseados somente na tarefa. A modelagem das metas de reutilização e a representação explícita de seus parâmetros também habilitam a melhoria contínua da performance da recuperação baseada no *feedback* da aplicação do método de recuperação na prática.

O objetivo da recuperação é trazer casos "úteis" relativo a uma meta específica a ser alcançada. Como a utilidade de experiências pode somente ser determinada quando estas forem reutilizadas como tentativa na situação atual, o critério de usabilidade a posteriori é previsto através do critério de similaridade entre a situação presente e a descrita na experiência, assumindo que situações similares (ou problemas) requerem soluções similares. Neste contexto, um assunto importante é a noção de similaridade. No domínio da mensuração de software, é muito improvável encontrar um GQM-PSEC na GQM-LL-KB que preenche completamente as necessidades da situação atual, porque cada produto de software, projeto e empresa são diferentes. Assim, tem-se que procurar preferencialmente por experiências que foram obtidas em situações passadas e que são similares a atual. Por exemplo, assumindo que

um engenheiro de garantia da qualidade queira reutilizar uma estratégia de solução para a seguinte situação descrita na Tabela 9 (Situação Atual).

Atributo	**Situação Atual**		**GQM-PSEC1**	**GQM-PSEC 2**	**GQM-PSEC 3**
setor de aplicação	automóvel		automóvel	automóvel	espaço aéreo
tamanho da equipe	10		15	100	50
descrição do problemas	O entrevistado não coopera com o entrevistador		O entrevistado não coopera com o entrevistador	Os resultados da entrevista não representam completamente o ponte de vista	Muitas metas identificadas
tarefa do problema	GQM2.2.2. Aquisição de conhecimento		GQM2.2.2. aquisição de conhecimento	GQM2.2.2.aquisição de conhecimento	GQM2.1. definição de metas

Tabela 9 Exemplo simplificado da recuperação do GQM-PSECs

Baseada em um conjunto de índices relevantes para a recuperação de experiências úteis, a descrição da situação atual é comparada com experiências armazenadas na GQM-LL-KB. Embora, seja bem improvável que uma GQM-PSEC com características idênticas seja encontrada, seja muito provável que no passado GQM-PSECs, a partir de "situações similares", foram relatadas. Por exemplo, projeto 1 com 15 desenvolvedores no setor de automóveis desenvolveu um programa de mensuração com um nível baixo de experiência, experimentando o mesmo problema durante a tarefa GQM 2.2.2 como mostrado na Tabela 9. Embora, não correspondendo perfeitamente as características do projeto atual, a solução aplicada no passado pode ainda ser utilizada como base para o problema atual. Então, a recuperação de experiências com características similares em relação a situação atual é necessária, ao invés de corresonder às características perfeitamente que, por exemplo, feito em um sistema de administração de base de dados tradicional ou sistemas de recuperação de informação. Essa não é uma tarefa trivial, como considera a avaliação a comparação de representações complexas de conhecimento para definir similaridade relativo a reutilização de lições aprendidas em mensuração.

A maioria dos sistemas usam índices para acelerar a recuperação de dados. Os índices são um subconjunto de atributos dos casos, que predizem a usabilidade do caso envolvendo a

descrição da situação dada. Ao invés de comparar atributos dos casos com a avaliação da situação atual durante a recuperação, somente os índices são comparados e utilizados para a determinação do valor de similaridade. Entretanto, os índices e sua relevância dependem de uma meta de reutilização específica e varia entre ambientes.

Com o objetivo de guiar a adaptação da metodologia a um ambiente específico, um método para a identificação de índices para uma meta de reutilização é desenvolvido [PG00] (ver Seção 9.2.2). O método guia a identificação inicial de um conjunto mínimo de índices por investigar e modelar caracterizações de contexto e analisar a estrutura do caso. O método também suporta a melhoria contínua e adaptação dos índices e mudanças no ambiente de aplicação.

Nas seções seguintes, descreve-se o método de recuperação em detalhes. Exemplos do método de recuperação orientado a metas para GQM-PSEC são demonstrados.

9.2.1. Processo de recuperação

Seguindo o método orientado a metas para a recuperação baseada em similaridade [GAB00,GAB99,Gre99] o processo de recuperação inclui os seguintes passos:

Passo 1. Avaliação da situação.

Primeiramente o usuário especifica a meta de reutilização, basicamente indicando o propósito do processo de reutilização (ex., prevenir ou resolver um problema). Dependendo da meta específica de reutilização, a situação é avaliada pela instanciação de um conjunto pré-definido de índices que descrevem a situação atual. Se disponível, parte dos índices requeridos pode ser deduzida a partir de informações no programa de mensuração em andamento. Por exemplo, características de contexto podem ser deduzidas da caracterização do projeto que é parte do plano do programa de mensuração. A importância de cada índice relativo a uma meta de recuperação é declarado através de um *fator de relevância* associado a cada índice. Fatores de relevância são armazenados na GQM-LL-KB e são automaticamente deduzidas em relação a uma meta de recuperação específica. Para facilitar a atribuição para índices desconhecidos ou considerados como irrelevantes em uma situação específica, nenhum valor necessita ser fornecido. A Tabela 10 ilustra a avaliação de uma situação com um conjunto exemplar de

índices. A avaliação da situação é apoiada também pelo conhecimento geral de domínio. Definições de tipo indicam intervalos válidos para valores de índices e auxiliam na aquisição de uma descrição consistente em projetos de software.

Passo 2. Correspondência exata entre índices marcados como essenciais. Num primeiro passo, os casos da GQM-LL-KB que correspondem perfeitamente com a avaliação da situação considerando os índices marcados como essenciais, determinam um conjunto de candidatos possíveis a reutilização. A Tabela 10 mostra um exemplo simplificado: enquanto compara casos da GQM-LL-KB com a avaliação da situação, o caso PSEC_003 e PSEC_011 são considerado como potenciais candidatos a reutilização, porque os valores do índices marcados como essenciais ("setor de aplicação" e "tarefa do problema") são iguais aos atuais. PSEC_007, que descreve uma experiência relativa a desenvolvimento de plano de mensuração, não é considerado, como o valor do índice "tarefa do problema" é diferente daquela de interesse.

Passo 3. Correspondência parcial de casos similares. Para todos os possíveis candidatos a reutilização um valor de similaridade é calculado pela correspondência parcial de índices (exceto aqueles marcados como essenciais) usando uma medida específica de similaridade relativa a meta (ver Seção 9.2). Casos com valor de similaridade maior que o limite dado são considerados como suficientemente similares e propostos ao usuário como candidatos a reutilização ordenados por seus valores de similaridade. Continuando o exemplo mostrado na Tabela 10, o caso PSEC_003 é considerado mais similar a situação dada que PSEC-007, porque os valores de índices marcados como importantes ou menos importantes (ex. "tamanho da equipe" e "descrição do problema") são mais similares com os atuais.

Passo 4. Seleção de candidato(s) a reutilização. Baseado nos candidatos propostos a reutilização o usuário pode selecionar o caso mais apropriado e, se necessário, adaptar manualmente o caso que melhor se adequa a suas necessidades. Decisões fundamentadas em informações são também suportadas pelas experiências capturadas explicitamente na GQM-LL-KB sobre as reutilizações de um caso particular no passado.

Meta de reutilização			**GQM Base de Conhecimento (resumo)**		
objeto	GQM-PSEC		CASO PSEC_003	CASO PSEC_007	CASE PSEC_011
propósito	guia da solução do problema				
processo	mensuração de software				
ponto de vista	equipe de garantia de qualidade				
ambiente	IntelliCar				
Índices					
departamento	irrelevante	ABS	FI	FI	FI
tamanho da equipe	menos importante	10	15	100	50
setor de aplicação	essencial	automóvel	automóvel	automóvel	automóvel
nível de experiência	importante	baixo	--	--	--
tarefa do problema	essencial	definição da meta de mensuração	definição de meta de mensuração	desenvolvimento de plano de mensuração	definição de meta de mensuração
descrição	importante	O entrevistado não coopera com o entrevistador	O entrevistado não coopera com o entrevistador	Resultado da entrevista não representam completamente o ponto de vista	Muitas metas foram identificadas

Tabela 10 Exemplo simplificado de recuperação

9.2.2. Identificação de índices

Visando adaptar a representação de GQM-PSECs a um ambiente particular um método para a caracterização e organização dos casos é desenvolvido [PG00]. O método auxilia a identificação de índices relevantes utilizado para a recuperação dos casos. Esses índices tem que ser escolhidos cuidadosamente considerando o ambiente específico e as metas de reutilização se os casos mais usados são a ser recuperados da GQM-LL-KB. Assim, tarefas e domínios a serem suportados necessitam ser analisados para encontrar descritores funcionalmente relevantes que devem ser usados para descrever e indexar GQM-PSECs. Bons índices são atributos e combinações de atributos que distinguem um caso de outros porque eles são indicações de alguma coisa importante naquele caso [Kol93]. Conseqüentemente, índices são aquelas combinações de atributos de um caso que descrevem as circunstâncias que

faz com que o caso seja considerado útil. [Kol93] propõe dois enfoques gerais para a escolha de índices:

- O enfoque funcional, que examina um conjunto de casos disponíveis e a tarefa a ser suportada, olhando para como cada caso pode ser usado e as formas que o caso precisa ser descrito para se tornar disponível.
- O enfoque da lembrança, que examina os tipos de lembranças que são naturais entre os especialistas humanos que executam a tarefa específica, procurando por similaridades relevantes entre novas situações e os casos recordados pelos especialistas, visando encontrar quais tipos de índices são importantes para o julgamento da similaridade e em quais circunstâncias.

Devido ao fato de que, quando uma Fábrica de Experiência é estabelecida na prática no domínio da Engenharia de Software, geralmente nenhum caso está disponível no início para guiar a caracterização e organização dos casos, a identificação de índices relevantes para a recuperação de PSECs é baseada no enfoque de lembrança.

O enfoque consiste de seis etapas [PG00]:

1. Identificação da meta de reutilização
2. Identificação de índices para contextualizar as experiências a serem reutilizadas
3. Identificação de características organizacionais
4. Coleta de informações sobre projetos realizados pela organização
5. Análise e interpretação dos dados coletados nas fases anteriores e definição de um conjunto características indexadoras da base de experiências
6. *Feedback* e aperfeiçoamento do conjunto de características encontradas

A abordagem cobre a caracterização e organização dos casos, bem como, a manutenção da representação dos casos. Nas seções seguintes, descreve-se cada etapa do método em detalhes.

9.2.2.1. Etapa 1: Identificação da meta de reutilização

O objetivo da etapa *Identificação da meta de reutilização* é identificar e caracterizar a(s) meta(s) de reutilização de experiências. A meta de reutilização corresponde a meta a ser

atingida através da utilização de um conjunto de experiências passadas em novas situações similares dentro da mesma organização. Uma meta de reutilização é composta por um conjunto de informações que após identificadas e analisadas, permitirão definir quais características/fatores possuem maior ou menor importância no processo de reutilização.

A definição da meta é realizada em reuniões para discussão de temas relacionados ao processo de reutilização das experiências de desenvolvimento de software na organização. A reunião tem como principal finalidade identificar as principais necessidades, problemas e objetivos dos vários tipos de pessoas envolvidas no processo. A partir disso, o engenheiro de conhecimento elaborará um plano para identificar metas de reutilização dos artefatos que comporão a base de casos. As metas de reutilização auxiliarão na identificação de características com maior ou menor relevância para o processo de reutilização.

As metas de reutilização seguirão um padrão semelhante a definição de metas de um plano GQM de mensuração, contendo:

Dimensão	**Definição**	**Tipos/Exemplos**
Objeto	O que será/poderá ser reutilizado? (Quais tipos de experiência farão parte da base de experiências)	Modelos de Processo Modelos de Produto Modelos de Qualidade Lições Aprendidas Dados Problemas/Soluções Material de Treinamento Manuais FAQ (*Frequently Asked Questions*) Outros
Objetivo	Para que o objeto será reutilizado?	Controle Prevenção de Falhas Melhoramento Planejamento Resolução de Problemas Avaliação Monitoramento Outros
Ponto de Vista	Quem irá utilizar as experiências no processo de reutilização?	Gerência Administrativa Gerência de Projeto Equipe de Projeto Grupo de Garantia de Qualidade Cliente Outro

Processo	Fase do desenvolvimento na qual as experiências vão ser reutilizadas?	Análise de Requisitos Projeto Implementação Manutenção Gerência de Configuração Testes Processo de Mensuração Processo de Desenvolvimento
Contexto	Em que ambiente serão reutilizadas as experiências?	Organização Departamento Área Setor Grupo

A meta de reutilização contém informações que apoiam a definição de metas a serem atingidas.

9.2.2.2. Etapa 2: Identificação de índices para estruturar as experiências

Índices podem recordar tanto o conteúdo quanto a especificação contextual. Especificações contextuais descrevem o ambiente no qual o episódio acontece, enquanto que o conteúdo especifica partes salientes que acontecem no episódio.

Baseado nas metas de reutilização identificadas, o conhecimento a ser recuperado tem que ser caracterizado considerando seu conteúdo. Isto pode incluir a descrição de um efeito preventivo, crença, tema, plano, resultados, efeitos colaterais, efeitos resultantes, etc. relacionados com o episódio que o caso descreve [Kol93]. Atributos preditivos para a respectiva meta de recuperação são determinados pelo engenheiro de conhecimento, que pode entrevistar especialistas sobre o ponto de vista declarado na meta de reutilização. Isso pode ser baseado também em um estudo de casos disponíveis descrevendo objetos a serem reutilizados.

9.2.2.3. Etapa 3: Identificação de características organizacionais

O objetivo da fase *Identificação de Características Organizacionais* é identificar características que descrevem o contexto organizacional, em organizações que tem como objetivo de negócio o desenvolvimento de software ou possuem áreas/equipes destinadas a desenvolver software. Nesta fase serão coletadas informações com o intuito de identificar aspectos que caracterizam a organização relacionadas ao processo de desenvolvimento de

software. Como uma base para a elaboração da informação um questionário para a caracterização da organização será aplicado pelo engenheiro de conhecimento com o gerente. Este questionário tem como objetivo capturar informações sobre o contexto organizacional onde são desenvolvidos os projetos de software.

Aspectos considerados no questionário incluem:

- Tipo da organização
- Setor industrial da empresa
- Estrutura Organizacional
- Atividades Desenvolvidas
- Área de Atuação
- Objetivo(s) de Negócio

9.2.2.4. Etapa 4: Coleta de informações sobre projetos realizados pela organização

O objetivo da etapa *Coleta de informações sobre projetos realizados pela organização* é coletar informações que descrevam os projetos de software realizados (ou em andamento) pela organização. As informações adquiridas nesta fase serão posteriormente analisadas (etapa 5) com o objetivo de discriminar características que diferenciem os projetos analisados e conseqüentemente forneçam índices potenciais para a base de casos. Como uma base para a elaboração da informação um questionário é aplicado. O questionário contém questões referentes aos projetos de desenvolvimento de software, características dos produtos desenvolvidos e recursos utilizados no processo de desenvolvimento. Características inicialmente não previstas no questionário e consideradas relevantes para os gerentes de projeto poderão ser anexados ao documento.

Aspectos considerados no questionário incluem:

- Quanto ao projeto:
  - Tipo
  - Natureza
  - Duração
  - Custos
  - Aspectos de Qualidade

  - Fatores de risco
  - Fatores de influência
- Quanto ao processo:
  - Modelo de Ciclo de Vida utilizado
  - Linguagem de programação
  - Metodologia de análise
  - Metodologia de testes
  - Estilo de documentação
  - Métodos de garantia de qualidade
- Quanto ao produto:
  - Área enfocada
  - Público alvo
  - Principais características (simplicidade, interface intuitiva, velocidade)
  - Plataforma

9.2.2.5. Etapa 5: Definição de um conjunto de índices para a base de experiências baseado na análise e interpretação dos dados coletados

Esta etapa foi subdividido em 3 partes:

- Definição de um conjunto inicial de características relevantes
- Identificação da relevância do conjunto de características iniciais baseado nas metas de reutilização
- Definição do conjunto final de índices

9.2.2.5.1.Etapa5.1: Definição de um conjunto inicial de características relevantes

O objetivo é analisar as informações coletadas das etapas 1 e 2 com o objetivo de selecionar um conjunto "geral" de características que representem as experiências que estarão contidas na base de casos.

Baseado num conjunto de questionários (com pelo menos duas caracterizações de projetos (questionário) e uma caracterização da organização), os dados dos questionários serão "cruzados" para verificar diferenças entre as respostas dadas. Onde existirem diferenças é

sinal de que a característica pode se tornar um possível índice para base de casos. Caso os vários projetos possuam sempre o mesmo valor para determinada característica, esta característica então será descartada como possível fator de indexação.

Primeiramente, com relação ao questionário sobre o contexto organizacional serão considerados possíveis índices as questões que tiverem como resposta mais de um item assinalado (ver Algoritmo 1).

SE ***número de itens assinalados*** = *1* ENTÃO
A questão é considerada sem relevância
SE ***número de itens assinalados*** > *1* ENTÃO
É um possível índice, nesse caso adicionar os itens assinalados na resposta como dentro da faixa de valores possíveis para o índice
FIM SE
Com relação ao questionário de projetos, as questões respondidas nos vários questionários entregues (pelo menos 2) serão analisadas mediante os seguintes critérios:
Considerando
R1 = Resposta da questão X do questionário 1
R2 = Resposta da questão X do questionário 2
...
Rn = Resposta da questão X do questionário n
SE ***R1*** = ***R2*** = ***Rn*** ENTÃO
A característica é considerada sem relevância
CASO CONTRÁRIO (alguma das resposta é diferente)
A característica é selecionada e os valores assinalados serão inseridos na faixa de valores possíveis para o índice
FIM SE

Algoritmo 1. Algoritmo para a definição de um conjunto inicial

Como resultado dessa fase, um conjunto geral de possíveis indexadores da base de experiências é determinado. Este conjunto servirá como uma base inicial de índices que será

refinada em conjunto final na etapa posterior.

9.2.2.5.2.Etapa 5.2: Identificação da Relevância do Conjunto de Características Iniciais Baseado nas Metas de Reutilização

O objetivo é identificar fatores de relevância para as características selecionadas na etapa 5.1 através da análise das metas de reutilização definidas na etapa 1. O passo será realizada uma reunião com a gerência de projeto juntamente com o engenheiro de conhecimento para que possa ser discutida a relevância das características selecionadas na etapa anterior. Cada meta será discutida individualmente, relacionando as características mais e menos relevantes para a determinada meta. O engenheiro de conhecimento através dos dados obtidos na reunião determinará os fatores de relevância de cada característica, baseado nas informações adquiridas durante a reunião. Estes fatores de relevância terão valores que variam segundo o critério abaixo:

0 – Sem relevância alguma;

1 – Pouca relevância;

2 – Relevante;

3 – Muito relevante;

4 – Essencial na caracterização das experiências.

O resultado dessa fase será um conjunto de características inicialmente determinada e seus pesos referentes a meta de reutilização.

9.2.2.5.3.Etapa 5.3: Definição do conjunto final de índices

O objetivo é obter um conjunto final de características (índices) que irão indexar a base de casos representando o contexto organizacional e os projetos desenvolvidos pela organização. O engenheiro de conhecimento, a partir, do conjunto geral de índices gerados na etapa 5.1 juntamente com os fatores de relevância levantados na etapa 5.2, irá analisar as informações como o objetivo de determinar um conjunto final de características para a base de

casos e os seus pesos correspondentes. As informações da etapa 5.2 permitirão ao engenheiro de conhecimento determinar quais características possuem maior ou menor relevância de acordo com a meta de reutilização abordada. A indicação dos fatores de relevância final será definido da seguinte forma:

- Para cada característica soma-se os pesos determinados na etapa 5.2 entre as várias metas de reutilização determinadas. O valor deste somatório é dividido pelo número de metas de reutilização.
- Os valores finais obtidos serão somados obtendo um valor total. Os valores individuais serão divididos pelo valor total. O resultado será um número entre 0-1 que corresponderá ao peso da característica.

Se o peso de uma característica for igual a 0 (zero), a característica é considerada sem relevância e será excluída do conjunto final de características.

O resultado dessa fase será o conjunto final de características indexadoras da base de experiências e seu respectivo fator de relevância.

9.2.2.6. *Feedback* e Aperfeiçoamento do Conjunto de Características Encontradas

O objetivo é melhorar continuamente o processo de determinação de características relevantes que discriminam experiências dentro de uma base de casos através de *feedback* fornecido pelos usuários da base de experiências durante o uso do sistema.

O sistema irá gerar arquivos de log para armazenar informações relevantes durante o processo de recuperação de casos. Estes arquivos serão analisados com o(s) objetivo(s) de:

- Identificar a quantidade de acessos aos índices utilizados na recuperação. Estes dados servirão para determinar os principais índices utilizados no processo de recuperação e também os índices que estão sendo pouco ou não utilizados.
- Identificar valores de atributos não relacionados na definição dos tipos.
- Identificar novas características não previamente relacionadas como características indexadoras de casos.
- Catalogar as situações em que o usuário não colocou um valor para determinada

característica.

- Protocolar quantas vezes o usuário reutilizou algum caso selecionado e qual caso foi reutilizado.
- Catalogar as situações em que o usuário não reutilizou nenhum caso.
- Protocolar as situações em que o usuário optou por reutilizar um caso menos similar.

Os arquivos de log permitirão que seja realizada uma análise sobre a utilidade dos índices determinados pela aplicação da metodologia.

Além disso algumas revisões nos índices poderão ser realizadas, normalmente seguindo 3 critérios:

- Periodicamente: em períodos pre-estabelecidos as características serão revistas (a partir dos dados coletados pelos arquivos de log) com o objetivo de verificar sua utilidade no processo de recuperação de casos;
- Mudança de ambiente: devido a alguma mudança na estrutura da organização, ou no processo de desenvolvimento de software ou nas metas de reutilização, o processo de determinação de índices deverá ser revisto para garantir a utilidade dos mesmos;
- Problemas com o sistema: se o sistema estiver apresentando resultados inadequados durante a recuperação de casos, então algum problema pode ter ocorrido na determinação dos índices. O processo de definição deverá ser revisado.

Após a análise dos dados coletados durante a utilização do sistema de recuperação, algumas ações poderão ser seguidas relacionadas aos problemas/situações que ocorram:

Situação	**Indicação**
Fator de relevância freqüentemente modificado	Rever o valor de relevância default
Freqüente uso de características não relacionadas	Revisão no processo de seleção de índices
Utilização de valores de atributos informados não catalogados no dicionário de dados	Adição do valor no dicionário de dados
Baixa utilização dos casos recuperados da base	Problemas com similaridade, revisar os critérios de similaridade adotados
Uso freqüente de um conjunto restrito de índices e pouquíssima utilização dos índices restantes	- Revisão do critério de determinação de índices - Verificar a possibilidade da exclusão de índices não utilizados

Frequente redefinição dos valores de pesos dos índices	Revisão no processo de determinação de índices, enfocando principalmente o dimensionamento dos pesos
Baixo acesso a base de casos ou Baixo número de casos reutilizado	- O processo de reutilização está retornando casos pouco similares - A base de casos está com poucos casos - Os casos da base não se encaixam no problema procurado - O usuário está com dificuldades com o uso do sistema
Alta utilização de casos com valores de similaridade menor que a do melhor caso (maior grau de similaridade)	Revisão nos critérios de similaridade entre casos

9.2.3. Determinação de similaridade

A determinação de similaridade pode ser feita pela aplicação de medidas de similaridade [Ric95]. Medidas de similaridade determinam para cada caso um valor numérico de similaridade e estima uma ordem parcial entre os casos na GQM-LL-KB. A similaridade dos casos pode ser julgada em dois níveis: global e local. *Medidas de similaridade global* são baseadas no número de índices que o caso passado e o caso atual têm em comum. No exemplo mostrado na Tabela 9, projeto 1 poderia ser considerado mais similar a situação atual (quatro índices correspondentes) que a situação 2 (dois índices correspondentes) ou situação 3 (nenhum índice correspondente). Considerando os requisitos específicos para a reutilização de GQM-PSECs no domínio da Engenharia de Software, utiliza-se um medida de similaridade flexível e parametrizável como definido em [GAB00,GAB99,Gre99] (ver Seção 9.2.3)

Além disso, dependendo da meta específica de recuperação, a correspondência entre certos índices pode ser mais importante que outros. Por exemplo, para a recuperação de lições aprendidas sobre o desenvolvimento de medidas, a linguagem de programação usada no projeto de software em estudo pode não ser importante, enquanto que a correspondência da meta de mensuração é decisiva e o nível de experiência é considerado importante. Assim, dependendo da meta de recuperação a relevância de cada índice para a determinação do valor similaridade global, tem que ser determinado e explicitamente representado por *fatores de relevância*. Os fatores de relevância especificam a importância de um certo índice com respeito a meta de reutilização. Os fatores relevantes são categorizados em termos de {essencial, importante, menos importante, irrelevante}. Se denotado como essencial, o

respectivo índice é utilizado para correnpondência perfeita no passo 2 do processo de recuperação. Por outro lado, o índice é associado a um peso usado para o cálculo da similaridade através de métodos de RBC [Gre99,Alt96].

Considerando somente a similaridade global pode-se frequentemente lidar com rejeição de casos como candidatos no domínio da ES, porque somente poucos valores de índices são idênticos ao a situação dada, embora dois casos possam ser bastante similares. Por exemplo, na procura por lições aprendidas em programação C, experiências relatadas na utilização de C++ podem também ser de interesse. Assim, a determinação de similaridade tem que ser estendida para considerar também *similaridade local* entre valores de índices. Então, medidas de similaridade local específicas tem que ser definidas para cada tipo de índice, ex., numérico, simbólico, ou textual em dependência ao ambiente específico. Sejam v_i e v_j dois valores de um atributo: v_i representa o valor da avaliação de uma situação dada e v_j o respectivo valor do caso na GQM-LL-KB. Então a similaridade local entre dois valores v_i, v_j é determinada através da medida de similaridade local $\upsilon'(v_i,v_j) \in [0,1]$ para o respectivo tipo. Medidas genéricas de similaridade local para os tipos básico W(v) (como definido em Seção 9.1.2) foram definidas em [GAB00,GAB99,Gre99], que podem ser adaptadas e refinadas dentro de um ambiente de aplicação específico.

O cálculo do valor local da similaridade é aprimorado pela integração de um tesauro (ver Seção 9.1.2). Isto permite a consideração de valores similares mas não necessariamente iguais para valores do tipo texto, por exemplo, se a descrição do problema refere-se ao "defeito" na situação atual e no caso da GQM-LL-KB é "falha" (um sinônimo para defeito nesse ambiente), então pelo uso do tesauro estes dois termos são considerados similares. Sem a integração do tesauro a similaridade entre os dois casos não seria identificada e, possivelmente casos relevantes não seriam observados para serem recuperados.

Frequentemente tem-se que lidar com informações incompletas considerando a avaliação da situação, bem como casos armazenados na GQM-LL-KB. Seguindo o método de recuperação orientado a metas [Gre00b], baseado no modelo de contraste Tversky [Tve77] como aplicado em PATDEX [Wes95], a similaridade de dois objetos é expressada através de contraste linear de diferenças de pesos entre suas características comuns e diferentes.

Considerando situações possíveis que podem ocorrer quando se compara a situação dada e o caso armazenado as seguintes *feature sets* são diferenciadas:

- E: Conjunto de características correspondentes. O valor da característica da dada situação corresponde aquela do caso armazenado. Por exemplo, se ambos, a situação dada e o caso armazenado declaram a característica "experiência do desenvolvedor" com alta.
- W: conjunto de características contraditórias. O valor da característica da situação dada não corresponde aquela do caso armazenado. Por exemplo, se no passado nenhuma ferramenta relatando esforço estava disponível, mas agora na situação dada a característica "ferramenta de controle de esforço" foi declarada com disponível.
- U: Conjunto de características desconhecidas. A característica esta presente somente no caso armazenado mas não está declarada na descrição da situação atual. Por exemplo, quando iniciado um projeto de software, certas informações como "tamanho do sistema" podem ter sido declaradas como desconhecida na descrição da situação, embora disponível em casos armazenados em projetos passados.
- R: Conjunto de características redundantes. A característica é declarada na descrição da situação dada, mas não contida no caso armazenado. Casos podem ter sido declarados incompletamente no passado por várias razões, ex., escassez de tempo ou falta de conhecimento do domínio. Por exemplo, durante o contínuo processo de aprendizado organizacional, a característica "experiência do desenvolvedor" pode não ter sido considerada inicialmente, mas posteriormente se tornou importante para a identificação de casos relevantes.

A subdivisão em diferentes conjuntos de características permite a manipulação explícita de informações desconhecidas e características correspondentes/contraditórias durante o processo de recuperação pela associação de um peso diferente a cada conjunto de características para cálculo do valor da medida global de similaridade.

Visando prevenir a sobrecarga do usuário com informação menos úteis, casos com poucos graus de similaridade são excluídos pela declaração de um *limiar global* que define quando casos são considerados suficientemente similares e um *limiar local* declarando quando dois valores de um atributo são considerados similares.

9.2.3.1. Medida de Similaridade

Considerando os requisitos específicos como descrito acima, a seguinte medida de similaridade sim(Sit',Ek') é utilizada para a determinação de valores de similaridade indicando o grau de similaridade entre um caso na GQM-LL-KB e a situação dada baseada no método de recuperação orientado a metas [GAB00,GAB99,Gre99].

A medida de similaridade é baseada nas seguintes hipóteses:

- Dada uma meta específica de reutilização g,
- Conjunto de índices $Cg=\{Cg_1, Cg_2,...\}$ relativos a meta de reutilização g
- $Sit'=\{(Cg_i, s_i) \in Sit \mid \text{fator de relevância } (S_i) \neq \text{essencial}\}$
- $caso_k = (E_k,e_k)$ da base de casos, com $E_k' \subseteq E_k \wedge \forall\ E_{ki}' \in Cg \wedge$ atributos $E_{ki}' \in Cg$ e seus respectivos valores e_{ki}',
- valor de similaridade local $\upsilon'(s_i,e_{ki}')$ do atributo s_i e e_{ki}' do respectivo tipo de atributo,
- peso ωg_i do vetor de relevância Rg relativo a meta g definindo um fator de relevância para a meta Cg_i,
- limiar local $\theta_i \in [0,1]$ para o índice Cg_i, definindo quando considerar dois valores similares,
- os conjuntos de características com pesos $\alpha, \beta, \gamma, \delta \in [0,1]$:

$E = \{Cg_i \mid (Cg_i \in Sit' \cap E_{ki}') \text{ e } (\upsilon'(s_i,e_{ki}') \geq \theta_i)\}$

$W = \{Cg_i \mid (Cg_i \in Sit' \cap E_{ki}') \text{ e } (\upsilon'(s_i,e_{ki}') < \theta_i)\}$

$U = \{Cg_i \mid (Cg_i \in E_{ki}' - Sit')\}$

$R = \{Cg_i \mid (Cg_i \in Sit' - E_{ki}')\}$

- A medida de similaridade global é definida como :

$$sim(Sit',Ek') = (\alpha \sum_{si \in E} \omega_{ik} \upsilon'(s_i,e_{ki}')) / ((\alpha \sum_{si \in E} \omega_{ik} \upsilon'(s_i,e_{ki}')) + (\beta \sum_{si \in W} \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))) + (\gamma \sum_{si \in U} \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))) + (\delta \sum_{si \in R} \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))))$$

$Case_k$ é considerado como candidato a reutilização, se todas as características marcadas como essenciais na situação dada correspondem exatamente as características do caso e

$sim(Sit', E_k') \geq \varepsilon$ limiar global.

9.2.4. Cenários de recuperação

Nas seções seguintes, o método de recuperação é detalhado para diferentes aplicações como identificado na Seção 3.2 e exemplos de parametrização do método de recuperação são apresentados. Entretanto, na aplicação do enfoque na prática a parametrização tem que ser revisada e se necessário adaptada ao ambiente específico.

9.2.4.1. Aplicação 1: Alerta a possíveis falhas

Antes de começar uma tarefa GQM (ex., antes de desenvolver entrevistas no passo GQM2.2.2 para a aquisição de conhecimento relevante relativo a meta de mensuração). O usuário pode requisitar uma visão geral sobre as falhas ocorridas em programas de mensuração passados, que foram originados nessa tarefa (ver Seção 3.2.1) com o objetivo de prevenir a repetição de problemas. Nesse caso a meta de reutilização é definida como mostrada na Tabela 11.

Modelo de meta	**Exemplo**
Recuperar <objeto>	GQM-PSEC
para o <propósito>	prevenção do problema
considerando <processo>	mensuração de software
a partir de <ponto de vista>	pessoal da garantia de qualidade
no contexto de <ambiente>	companhia IntelliCar

Tabela 11 Exemplo de meta de reutilização

Considerando a respectiva meta de reutilização, os parâmetros do método de recuperação [GAB00,GAB99] são definidos como apresentado em Tabela 12.

A situação atual é descrita por características de contexto relevantes (ex., nome da empresa, setor de aplicação) e a tarefa de interesse, ex., aquisição de conhecimento (GQM2.2.2). Candidatos a reutilização são identificados por corresponder exatamente a tarefa

Atributo	Descrição
conjunto de índices	organização, tamanho da equipe, setor de aplicação, setor de aplicação, tarefa GQM
vetor de fatores de relevância	(organização-0.08); (tamanho da equipe-0.08); (setor de aplicação-0.14); (setor de aplicação-0.7); (tarefa GQM-essencial)
conjuntos dos pesos de *feature sets*	$\alpha = 1.0$; $\beta = 0.1$; $\gamma = 0$; $\delta = 0.1$
limiar de similaridade global	$\varepsilon_k = 0.3$
números de casos apresentados	20
saída	problema GQM-PSEC

Tabela 12 Exemplo de parametrização do método de recuperação

de interesse (marcado como essencial - como o usuário está interessado somente em problemas considerando uma determinada tarefa) com os casos na GQM-LL-KB e corresponde parcialmente das características de contexto dadas. Por exemplo, dado um setor de aplicação específico como escopo de interesse, experiências obtidas em um setor particular são considerados mais similares a situação atual que experiências obtidas em outros setores de aplicação. Pela instanciação da medida de similaridade global com os parâmetros definido (ver Tabela 12) um valor de similaridade é determinado para os casos na etapa 3 do processo de recuperação. Os 20 casos mais similares ordenados por seus valores de similaridade, sendo o valor de similaridade maior ou igual ao limiar de similaridade global definido ($\varepsilon_k = 0.3$) são considerados como candidatos potenciais a reutilização. Como o objetivo do pedido de recuperação é fornecer uma visão geral de possíveis falhas originadas na tarefa GQM de interesse, o problema é extraído dos casos recuperados e apresentados ao usuário.

Além disso o GQM-PSEC completo pode ser explorado pelo usuário, por exemplo, examinando a respectiva caracterização de contexto indicando o contexto válido.

9.2.4.2. Aplicação 2 : Guia para a solução de problemas

Quando problemas específicos ocorrem, o usuário pode requisitar auxílio para guiar a solução pela reutilização de soluções de problemas similares do passado (ver Seção 3.2.2). Assumindo, por exemplo, que durante o passo GQM 2.2.4, não foi possível definir completamente um modelo de qualidade. Nesse caso a meta de reutilização é definida como

mostra a Tabela 13.

Modelo de meta	**Exemplo**
Recuperar <objeto>	GQM-PSEC
para o <propósito>	resolver o problema
considerando <processo>	mensuração de software
a partir de <ponto de vista>	equipe de garantia da qualidade
no contexto de <ambiente>	companhia IntelliCar

Tabela 13 Exemplo de meta de reutilização

Considerando a respectiva meta de reutilização, os parâmetro do método de recuperação [GAB00,GAB99] são definidos como apresentado em Tabela 14.

Atributo	**Descrição**
conjunto de índices	organização, tamanho da equipe, setor de aplicação, nível de experiência, tarefa do problema, descrição do problema, objeto do problema, função do problema, meta não atingida
vetor de fatores de relevância	(organização-0.04); (tamanho da equipe-0.04); (setor de aplicação-0.06); (nível de experiência-0.08); (tarefa do problema-essencial); (descrição do problema-0.5); (objeto do problema-0.2); função do problema-0.1); (meta não atingida-0.08)
pesos dos conjuntos de *feature sets*	$\alpha = 1.0$; $\beta = 0.1$; $\gamma = 0$; $\delta = 0.1$
limiar de similaridade global	$\varepsilon_k = 0.5$
número de casos apresentados	5
saída	solução GQM-PSEC

Tabela 14 Exemplo de parametrização de método de recuperação

A situação atual é descrita através do problema (por exemplo, descrição do problema, tarefa do problema) e características de contexto relevantes. Candidatos a reutilização são identificados pela correspondência exata da tarefa do problema (marcados como essencial) com os casos na GQM-LL-KB e ccorrespondendo parcialmente com as características de contexto dadas e os índices relacionados ao problema. Pela instanciação da medida de similaridade com o parâmetro definido (ver Tabela 14) um valor de similaridade é determinado para os casos na etapa 3 do processo de recuperação. Os 5 casos mais similares

ordenados pelo seus valores de similaridade, sendo o valor de similaridade maior ou igual ao limiar de similaridade global (ε_k = 0.5) são considerados como potenciais candidatos a reutilização. Os candidatos possíveis a reutilização são fornecidos ao usuário, apresentando informação sobre a solução aplicada em casos passados. O usuário pode também explorar os casos recuperados, (ex. informação de reuso) com o objetivo de fazer uma decisão fundamentada em informações. Se necessário, soluções sugeridas são adaptadas pelo usuário visando atender as necessidades atuais.

9.3 Aquisição de lições aprendidas em mensuração

Nesta seção será descrito um método para aquisição de lições aprendidas a partir de um programa de mensuração em andamento na empresa.

A evolução incremental da GQM-LL-KB é essencial. Conseqüentemente, o conhecimento na GQM-LL-KB tem que ser melhorado e atualizado cada vez que um novo programa de mensuração é planejado. Essa não é uma tarefa trivial, pois necessita a captura de conhecimento tácito de como aplicar planejamento de software de mensuração. Nesse contexto, segue-se o a abordagem RBC enfocando primeiramente a aquisição de casos descrevendo um problema em particular que aconteceu no passado e como foi resolvido. Cada vez que novas lições relativas ao planejamento de programas de mensuração são aprendidas, estas são incluídas na GQM-LL-KB. Espera-se melhorar o conhecimento representado na GQM-LL-KB e adaptá-la as mudanças de ambiente. Assim, cada vez que lições aprendidas em mensuração são reutilizadas para suportar o planejamento de programas de mensuração, experiências novamente obtidas no programa de mensuração atual são capturadas. Visando manter um mínimo de esforço em aquisição de conhecimento, esse processo tem que ser mesclado ao próprio processo de reutilização.

Considerando a aplicação 2 (ver Seção 3.2.2), quando na requisição de uma solução de um problema ocorrido atualmente, a possibilidade de capturar uma nova lição acontece, através da descrição do problema atual e como será resolvido. Em contraste, aplicação 1, a situação em que ainda não ocorreu um problema, não fornece possibilidade de obter um novo caso. Assim, o foco da aquisição de conhecimento é em aplicações do tipo 2, onde o processo

de recuperação de soluções de problemas é usado em paralelo para adquirir um novo episódio de solução de problema. Informações fornecidas pelo usuário como entrada ao processo de recuperação e experiências recuperadas da GQM-LL-KB são reutilizados durante a aquisição com o objetivo de otimizar esse processo.

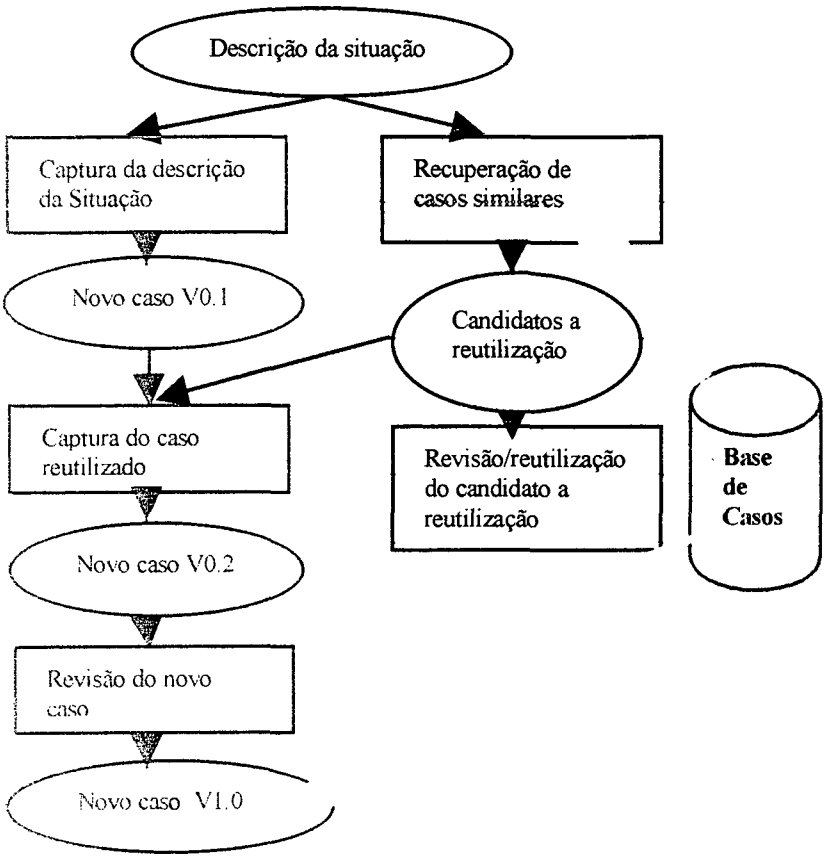


Figura 9 Processo de aquisição

Quando o usuário requisita a recuperação de GQM-PSECs úteis para resolver o problema atual, ele fornece uma avaliação inicial da situação atual. Por exemplo, quando reutiliza GQM-PSECs visando suportar a solução de um problema encontrado considerando a definição do modelo de qualidade para uma questão de plano GQM: " Qual é a distribuição de defeitos?", o usuário fornece as seguintes informações de contexto: "organização: IntelliCar, domínio de aplicação: automóvel". Em paralelo visando usar essa informação dada como entrada para o processo de recuperação, essa informação é armazenada em um novo caso documentando o episódio do problema atual (ver Figura 9).

Como resultado, o esforço de aquisição é reduzido, como a descrição separada de contexto e o problema para a documentação do novo caso não é mais necessária. Uma vez que GQM-PSECs similares são recuperados e um dos candidatos a reutilização é selecionado pelo usuário como guia para a solução do problema atual, informações contidas no GQM-PSEC reutilizado (ex., causa, solução e resultado) são usadas para complementar a documentação do novo caso (ver Figura 10). A cópia de informações do caso reutilizado pode também reduzir esforços, com as informações sobre a causa, solução e resultados não tem que ser descritas desde o início para a nova situação.

Quando se captura um novo caso, este tem que ser melhorado com informações básicas (ver Seção 9.1.1) ex., data de criação e representatividade. Aqui, informações que não podem ser deduzidas tem que ser fornecidas pelo usuário. Além disso, se um caso da GQM-LL-KB foi reutilizado, suas informações sobre reutilização tem que ser atualizadas pela obtenção das respectivas informações pelo usuário.

A documentação derivada semi-automaticamente do novo caso tem que ser cuidadosamente revista pelo usuário. Informações que estejam faltando tem que ser adicionadas e desvios na informação copiada do caso reutilizado ter que ser modificados, p.ex., se uma solução daquela declarada no caso reutilizado foi usada. Um exemplo simplificado de um GQM-PSEC adquirido é apresentada na Figura 10.

A aquisição manual de novos GQM-PSECs é guiada através de uma estrutura de caso detalhada, que indica explicitamente dimensões relevantes a serem capturadas O uso do conhecimento geral de domínio em forma de definições de tipo facilita também uma descrição consistente de experiências em projetos de software e programas de mensuração.

9.4 Integração de novas lições aprendidas em mensuração

Os novos GQM-PSECs adquiridos tem que ser integrados na GQM-LL-KB existente e domínio geral de conhecimento tem que ser extraído para estar efetivamente disponível para uso futuro. Além disso, a performance da GQM-LL-KB pode ser continuamente melhorada baseada no *feedback* obtido pela sua aplicação na prática.

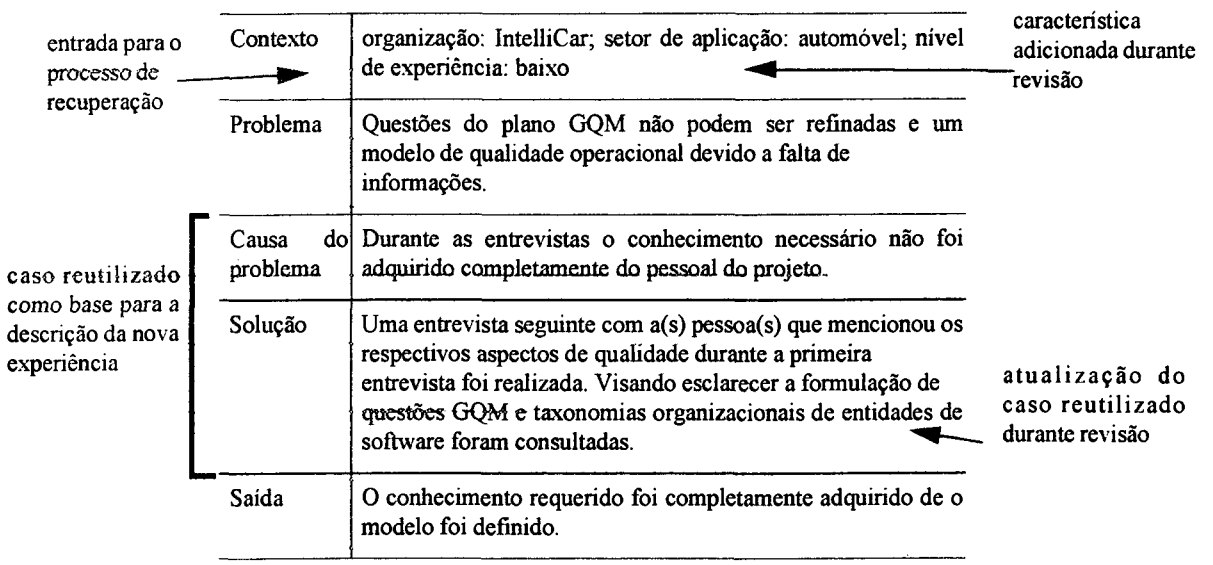


Figura 10 Exemplo simplificado de aquisição de GQM-PSEC

As técnicas para a integração apropriada são descritas nas seguintes seções em detalhes.

9.4.1. Integração de GQM-PSECs

A integração de novos casos, em forma de GQM-PSECs, representa o processo de aprendizado. Isto implica que casos tem que ser armazenados, interdependências tem que ser criadas ou adaptadas e, se necessário, padrões genéricos de casos tem que ser criados ou modificados. O armazenamento de casos inclui a seleção de casos a serem incluídos, a garantia de qualidade da informação fornecida em casos e sua representação apropriada na GQM-LL-KB.

O foco da GQM-LL-KB é primeiramente armazenar lições aprendidas concretas referenciando uma situação de problema específica. Entretanto, se um novo caso é uma cópia e difere somente em detalhes de um caso armazenado na GQM-LL-KB, um caso abstrato substituindo os dois casos de projeto específico pode ser criado através de generalização de caso [BW96]. O desenvolvimento de padrões genéricos pelo engenheiro de conhecimento pode ser guiada por taxonomia relacionadas, como especificado nas definições de tipos, que

fornecem uma base para derivação de novas abstrações substituindo valores específicos.

9.4.2. Atualização e melhoria do conhecimento geral de domínio

A atualização e melhoria do conhecimento geral de domínio na GQM-LL-KB são também associadas ao processo de integração. Definições de tipo são revisadas e atualizadas baseadas no *feedback* obtido de um novo caso utilizado. Por exemplo, se o intervalo de valores do atributo "função do problema" foi definida como {engenheiro de garantia de qualidade, desenvolvedor de software, testador}, mas no novo caso é especificado como "gerente de processo", a nova função é incluída no intervalo de possíveis valores após a revisar sua relevância no ambiente específico. A terminologia usada para a descrição do caso tem que ser cuidadosamente revista considerando o ambiente específico. Se termos são inconsistentemente usados, p.ex. utilizando os termos de desenvolvedor de software e engenheiro de software para a mesma função, terminologia apropriada tem que ser determinada. Se termos são usado como sinônimos em um ambiente específico, os termos são incluídos no tesauro.

9.4.3. Melhoria da performance da recuperação

A contínua evolução e adaptação da GQM-LL-KB a um ambiente específico pode também necessitar a modificação da representação de GQM-PSECs ou de parâmetros para a recuperação da GQM-LL-KB na prática.

Devido ao fato de que índices dependem do ambiente específico e podem mudar ao longo do tempo, a contínua adaptação do esquema de índices necessita ser suportada durante todo o ciclo de vida da GQM-LL-KB pelo engenheiro de conhecimento. Por exemplo, complementares características de contexto de projetos de software podem se tornar relevantes para a descriminação de casos. Como mostrado na Figura 10, o atributo "maturidade da mensuração" não foi considerada como uma característica relevante para a descrição de contexto de um caso passado, porque experiências foram relacionadas a projetos sem variações considerando maturidade com respeito a mensuração de software.

Uma vez que um novo programa de mensuração é estabelecido no projeto com um

diferente nível de maturidade, esse atributo se torna relevante para a distinção de casos e é adicionado a caracterização de contexto. *Feedback* na aplicação do enfoque na prática pode ser usado como base para atualizar o esquema de índice relativo a uma meta específica de reutilização. Por exemplo, se frequentemente valores não são fornecidos para um índice, esse índice pode ser irrelevante para o processo de correspondência. Além disso, através da análise de mudanças que acontecem no ambiente e entrevista com o usuário, potencial de melhoria também pode ser identificado, tal com a necessidade de inclui um novo índice.

Contínuo aprendizado tem também ocorrido considerando a medida de similaridade e sua parametrização para metas de reutilização específicas visando melhorar e otimizar sua performance. Assim, o processo de recuperação e reutilização pode ser supervisado e, baseado no *feedback*, adaptado apropriadamente ao ambiente específico pelo engenheiro de conhecimento. Exemplo de aplicações de melhorias necessárias são:

- frequentemente não são fornecidos valores para um índice -> mudar o fator de relevância atribuído ao índice.
- número de candidatos recuperados aumenta -> mudar a estratégia otimista para medida de similaridade em estratégia mais pessimista ou aumentar o limiar.
- frequente rejeição de casos sugeridos como candidatos a reutilização -> se uma meta de reutilização específica é afetada, então aumentar o limiar global relacionado com a meta, ou, revisar o esquema de indexação e medida de similaridade considerando críticas e sugestões adicionais do usuário.

Baseado na análise cuidadosa de causas, a seleção de índices e/ou medidas de similaridade tem que ser adaptadas visando melhorar os resultados de recuperação no futuro.

O processo de integração é basicamente desenvolvido pelo engenheiro de conhecimento. Se necessário, a equipe do projeto ou da garantia da qualidade está envolvido na coleta de *feedback* e na resolução de conflitos e inconsistências para garantir a qualidade de casos armazenados na GQM-LL-KB.

10 Protótipo de Ferramenta GQM-LL

Esta seção fornece uma visão geral do protótipo de ferramenta GQM-LL, descrevendo sua arquitetura e funcionalidade em cenários de aplicação básicos.

10.1 Arquitetura

A ferramenta GQM-LL é um protótipo altamente integrado no ambiente da ferramenta de mensuração REMEX [RG00,Gre99]. REMEX é um ambiente compreensivo que fornece suporte para o processo de planejamento inteiro para a edição e documentação de produtos GQM. Além disso, fornece facilidades para reutilização de produtos GQM, bem como lições aprendidas (através da integração da ferramenta GQM-LL) para todas as tarefas do processo de planejamento.

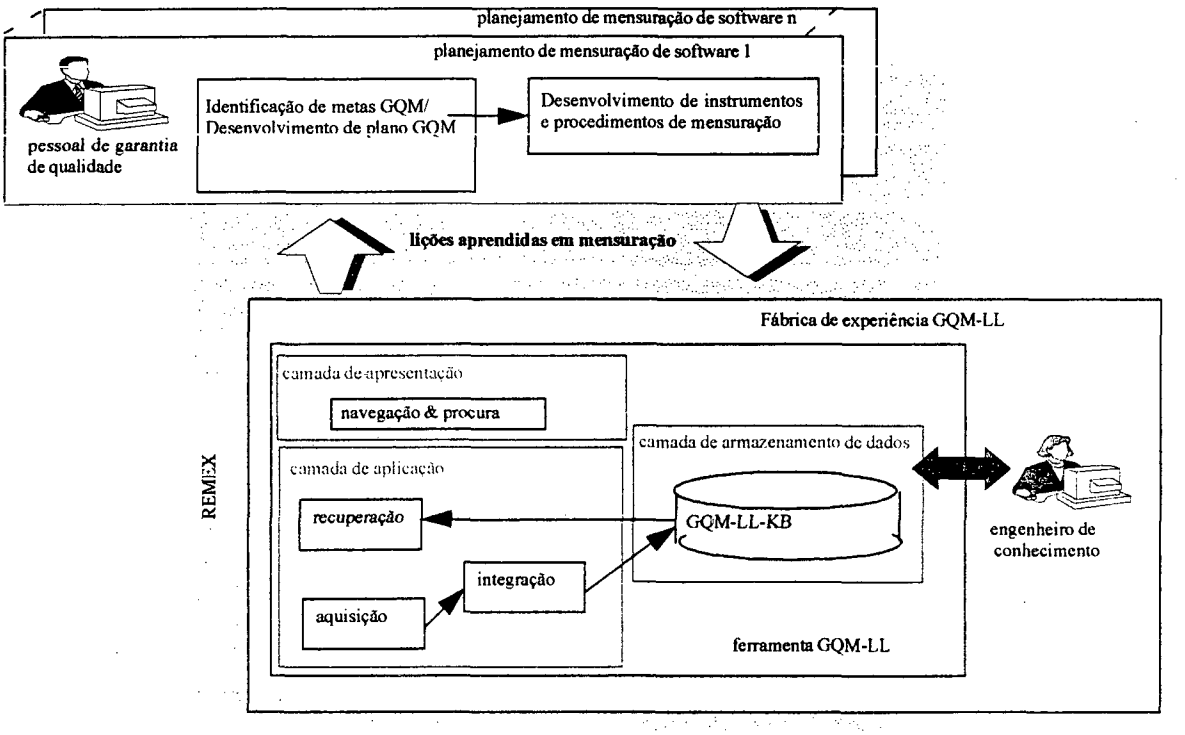


Figura 11 Arquitetura e integração da ferramenta GQM-LL

A ferramenta é basicamente uma arquitetura cliente/servidor consistindo de três camadas lógicas: apresentação, aplicação e armazenamento de dados (ver Figura 11). O conhecimento, incluindo GQM-PSECs bem como o conhecimento geral de domínio é armazenado na GQM-LL-KB. A camada de aplicação fornece ferramentas de suporte para a recuperação de GQM-PSECs e a aquisição e integração de novas lições aprendidas. O objetivo da ferramenta GQM-LL é mais ser um assistente inteligente integrado ao ambiente de mensuração do que fornecer suporte automatizado. Assim, ferramentas para a apresentação de conhecimento ao usuário durante a recuperação e aquisição estão disponíveis, bem como ferramentas de manutenção apoiando o engenheiro de conhecimento para a integração de novas experiências na GQM-LL-KB existente.

O protótipo da ferramenta GQM-LL foi desenvolvido em Smalltalk independente de plataforma usando o VisualWorks 3.0.

10.2 Instanciação inicial de aplicação específica a organização

Durante a aplicação da ferramenta GQM-LL pela primeira vez em uma organização em particular, certos parâmetros têm que ser revisados e definidos. Isto inclui a determinação de índices relativos à caracterização de contexto, a definição inicial do conhecimento geral de domínio e parâmetros de recuperação.

Usando o método para a caracterização e organização de experiências, um conjunto mínimo de índices relevantes para a identificação de GQM-PSECc relevantes e diferenciais em um ambiente pode ser determinado.

O conhecimento geral de domínio específico a um ambiente particular tem que ser inicialmente definido. Essa tarefa inclui a definição de tipos para os novos índices, bem como a revisão de tipos de índices pré-definidos. Além disso, sinônimos já conhecidos usados frequentemente na organização podem ser incluídos no tesauro de mensuração.

Visando adaptar o processo de recuperação a um ambiente específico, o conjunto de parâmetros de recuperação pré-definido (incluindo metas de reutilização, índices e medidas de

similaridade) tem que ser revisados, e se necessário, adequados propriamente.

10.3 Processo de recuperação

Nesta seção, aplicações de exemplos da ferramenta GQM são apresentados para dois cenários de aplicação básicos (ver Seção 3.2).

10.3.1. Cenário1: Aviso de possíveis falhas

Assumindo que pela primeira vez um programa de mensuração é estabelecido no departamento ABS da companhia IntelliCar e uma nova equipe de garantia da qualidade foi alocada para a introdução de mensuração neste departamento. Durante o desenvolvimento do plano GQM, entrevistas tem que ser desenvolvidas visando adquirir todas as informações relevantes relativas à meta.

Esta é uma tarefa crítica e custosa do programa de mensuração, pois envolve a participação da equipe do projeto e o conhecimento adquirido serve como base para o programa de mensuração completo. Assim, os dois engenheiros de garantia de qualidade no cargo requisitam, antes de iniciar a série de entrevistas, todas a lições aprendidas disponíveis

sobre problemas que o ocorreram durante essa tarefa em programas de mensuração passados.

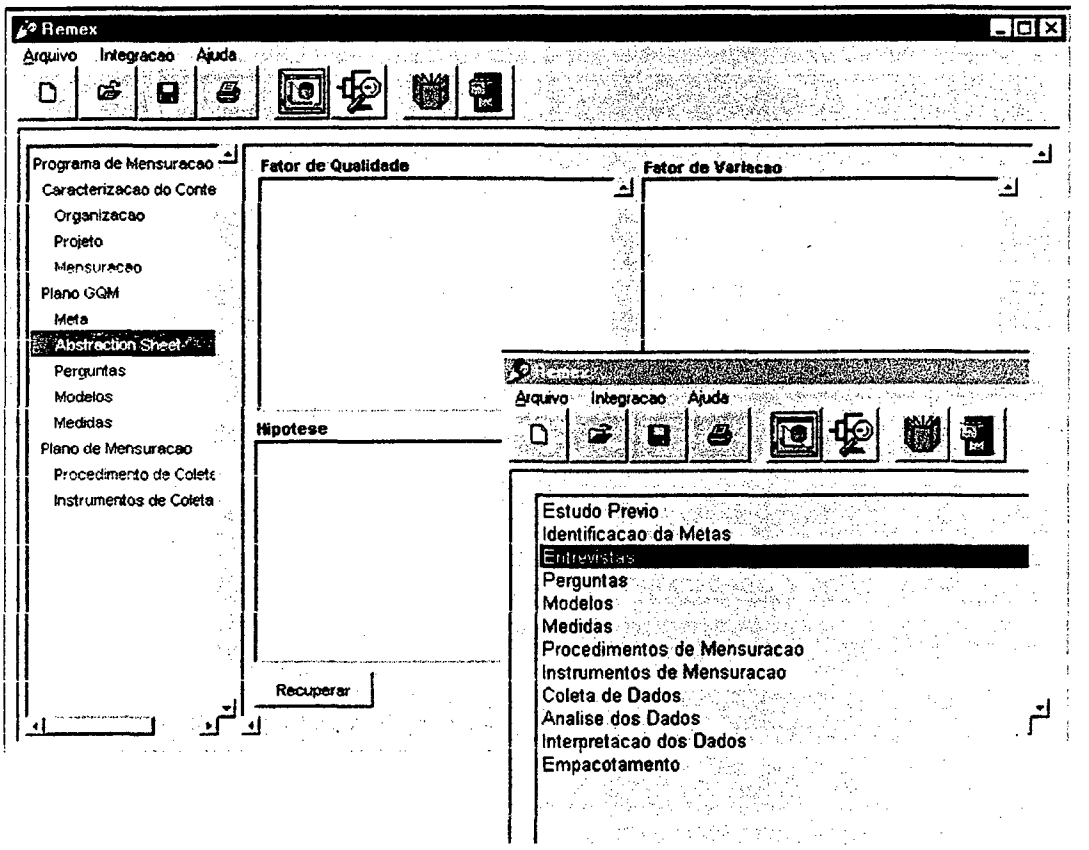


Figura 12 Exemplo de requisição de recuperação

Assim, o engenheiro de garantia da qualidade seleciona a tarefa "aquisição de conhecimento" visando recuperar experiências relevantes. Outra informação de entrada para o processo de recuperação, tais como setor de aplicação, etc., é automaticamente deduzida de informações disponíveis no programa de mensuração.

Por exemplo, o contexto do programa de mensuração foi definido no passo GQM 1 durante o estudo prévio e documentado no plano de mensuração. Como essas informações já estão disponíveis na base de conhecimento, as respectivas informações não necessitam ser

fornecidas manualmente pelo usuário.

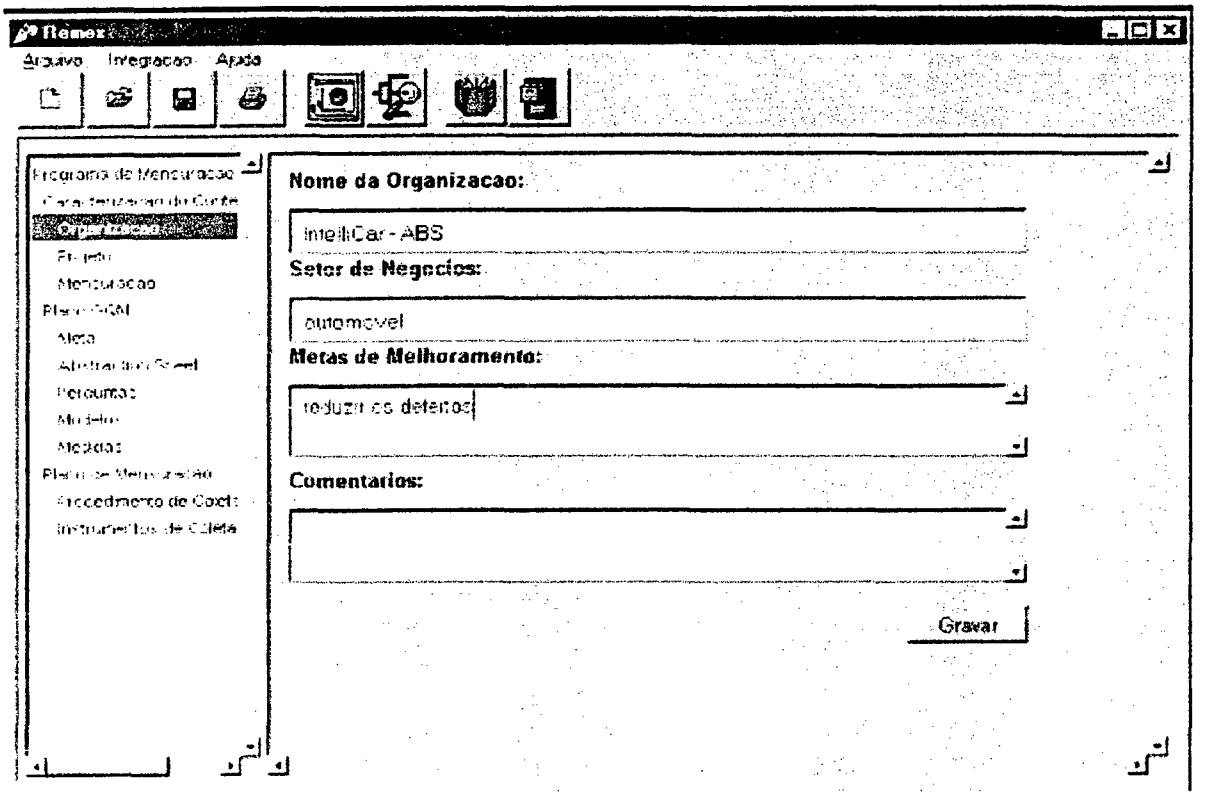


Figura 13 Exemplo de caracterização da organização

Durante a instanciação do método de recuperação com parâmetros da meta de recuperação definidos, a ferramenta GQM-LL recupera uma lista de problemas relatados a

tarefa específica de planejamento como resultado do pedido de recuperação.

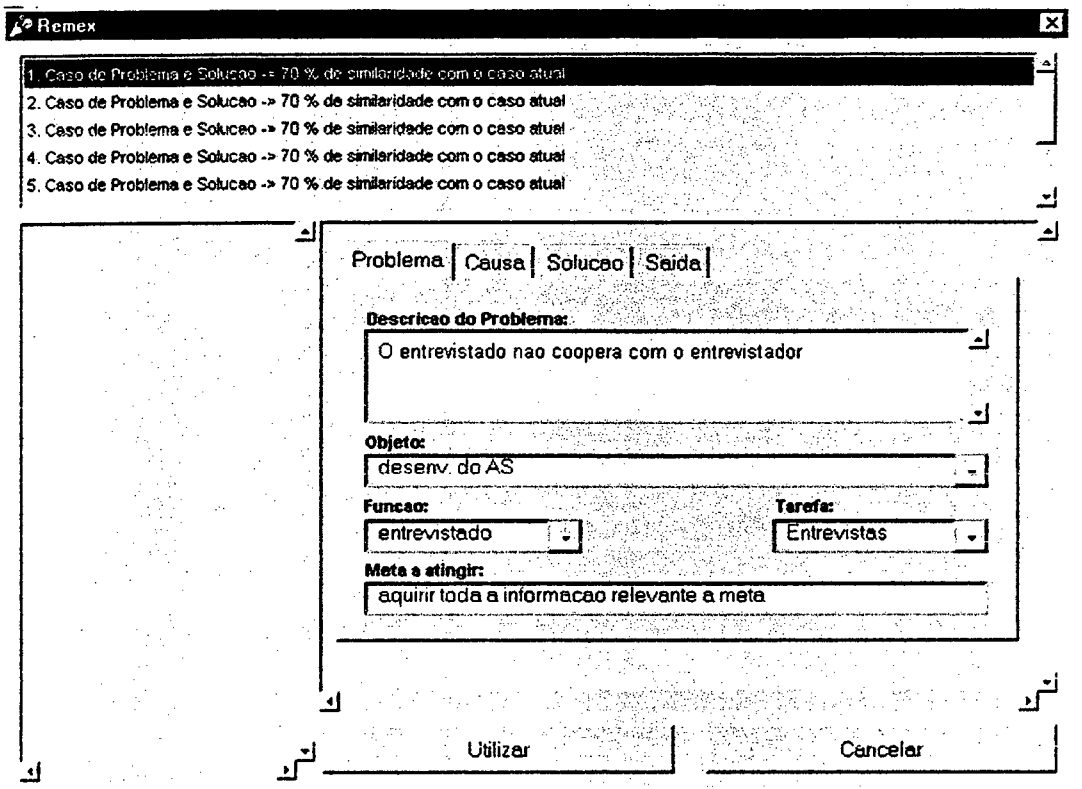


Figura 14 Exemplo de resultado de recuperação

Agora, os candidatos a reutilizaçao podem ser examinados pelos engenheiros de garantia de qualidade (ver Figura 14). Se necessário, informações adicionais sobre os casos recuperados podem também ser exploradas. Por exemplo, o engenheiro de garantia de qualidade pode também ser informado sobre o problema declarado no caso, investigando as causas do problema.

10.3.2. Cenário 2: Guia para a solução de um problema

Assumindo que, durante o desenvolvimento de modelos de qualidade no programa de mensuração no departamento ABS, um problema ocorreu: *No atual programa de mensuração um modelo de qualidade sobre a distribuição de defeitos por criticalidade não pode ser completamente definida, como as categorias para criticalidade de defeitos eram desconhecidas*. Nesse caso, o engenheiro de garantia da qualidade pode investigar GQM-

PSECs com o objetivo de encontrar uma estratégia adequada para resolver o problema.

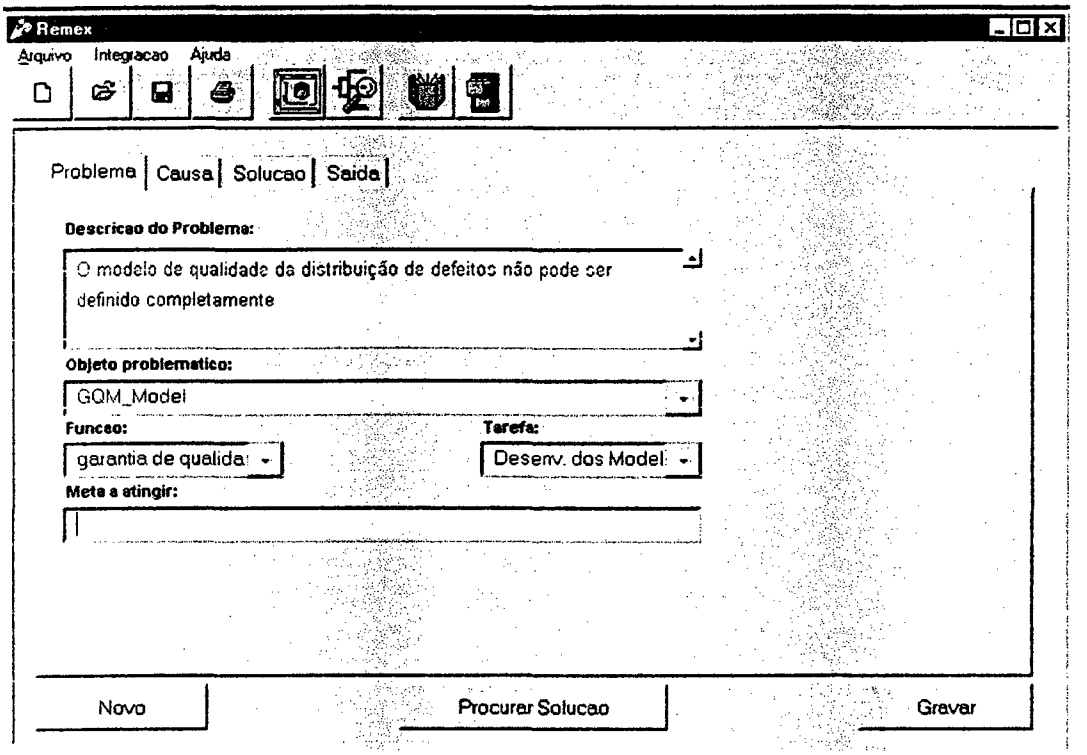


Figura 15 Exemplo de pedido de recuperação

Desse modo, o engenheiro de garantia de qualidade descreve o problema ocorrido atualmente em termos de atributos declarados na interface do problema (ver Figura 15). Se quaisquer dos atributos são considerados irrelevantes ou seus valores são desconhecidos na situação específica, nenhum valor necessita ser fornecido. A descrição da situação atual é apoiada também através das definições de tipo dos respectivos atributos, indicando possíveis valores de atributos, por exemplo, a função envolvida no respectivo problema. Se necessário, esses valores podem ser modificados dinamicamente pelo usuário. Informações adicionais relativas a caracterização de contexto são automaticamente deduzidas da documentação do plano do programa de mensuração como na aplicação 1 (ver Figura 13).

Com base na avaliação da situação ilustrada e no método de recuperação instanciado com os parâmetros relativos a respectiva meta de reutilização, GQM-PSECs similares são recuperados e apresentados ao usuário ordenados por seus graus de similaridade (ver Figura 16).

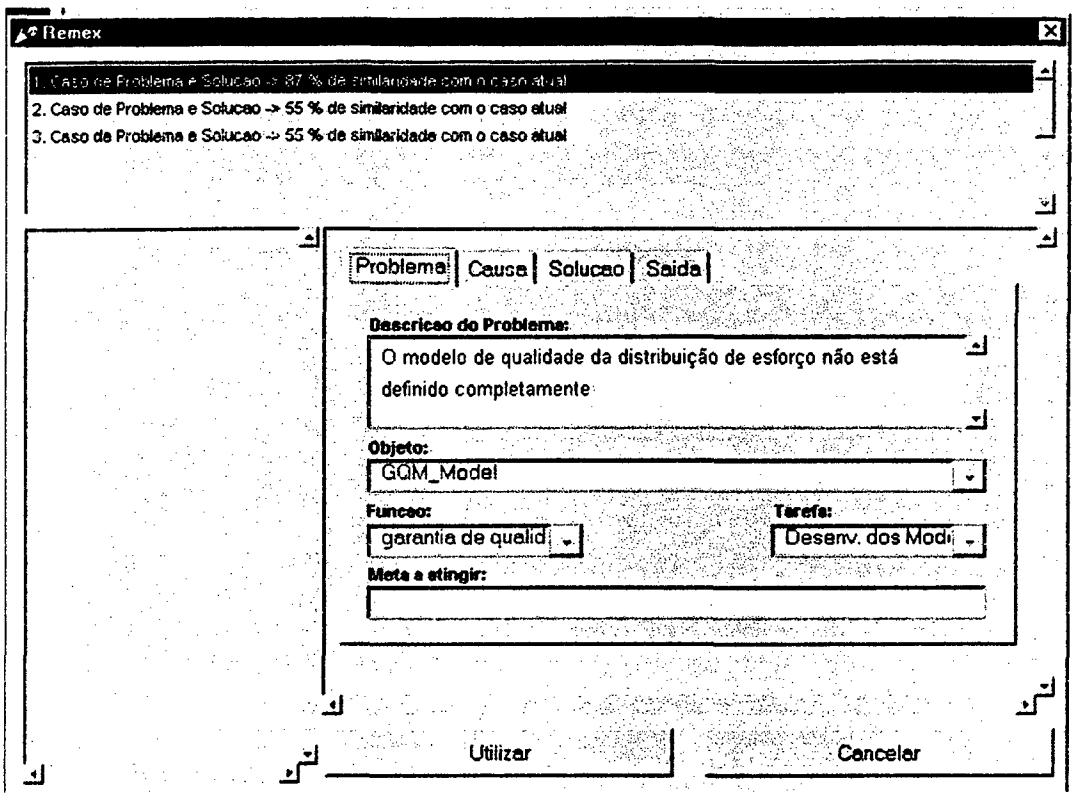


Figura 16 Exemplo de resultado de recuperação

Agora os engenheiros de garantia da qualidade podem investigar as estratégias de solução adotadas no passado para resolver problemas similares. Outras informações sobre os respectivos casos, ex., a causa do problema ou as saídas obtidas pela aplicação da solução

podem ser exploradas visando guiar decisões fundamentadas para as estratégias de solução.

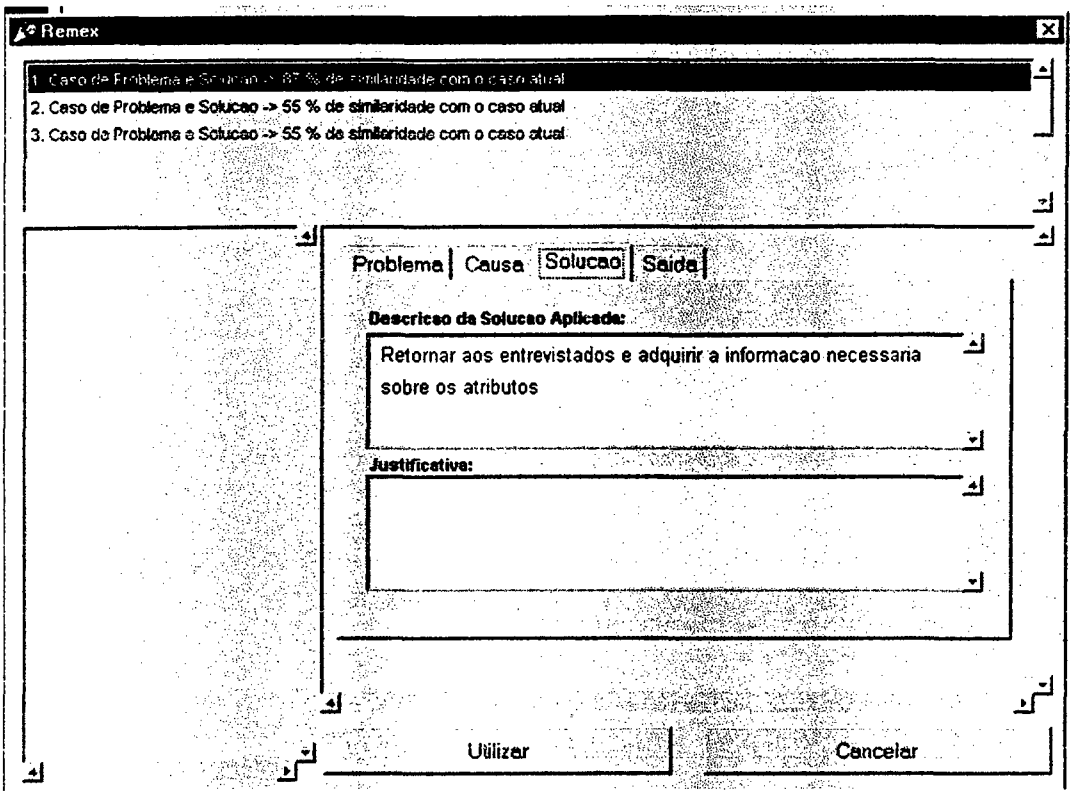


Figura 17 Exemplo de informação da solução do caso recuperado

10.4 Processo de aquisição

A aquisição de um novo caso é integrada ao processo de recuperação da aplicação 2. Quando se descreve o problema específico na presente situação como entrada para o processo de recuperação (ver Seção 9.2.4), a informação é capturada em paralelo para a documentação do novo GQM-PSEC descrevendo o novo problema. Uma vez que um candidato a reutilização recuperado é selecionado para reutilização, suas informações relativas a causa, solução e saída são copiadas na documentação do novo caso. O novo caso é criado dessa maneira e então é revisado pelos engenheiros de garantia de qualidade antes de ser capturado pela ferramenta GQM-LL. Necessariamente modificações e adições ao caso criado tem que ser feitas pelos engenheiros de garantia de qualidade visando relatar corretamente o caso presente.

10.5 Processo de integração

A integração de novos casos na base de experiências é uma tarefa complexa. Então, o foco da ferramenta é mais fornecer uma assistência inteligente ao engenheiro de conhecimento que é responsável pela integração adequada de novos casos. Isto inclui basicamente a integração de GQM-PSECs novamente adquiridos, atualização e melhoria do conhecimento geral de domínio e melhoria da performance da recuperação baseada no *feedback* de sua aplicação.

Esses passos são descritos nas seções seguintes.

10.5.1. Integração de GQM-PSECs

Os GQM-PSECs novos adquiridos têm que ser integrados na GQM-LL-KB existente. Assim, a ferramenta fornece uma lista de todos os novos casos capturados. Antes de serem armazenados permanentemente na GQM-LL-KB, os casos têm que ser revisados pelo engenheiro de conhecimento (ver Seção 9.3) . Uma vez aceitos pelo engenheiro de conhecimento, os casos são transferidos e armazenados na GQM-LL-KB onde se tornam disponíveis para uso.

10.5.2. Atualização e melhoria do conhecimento geral de domínio

Parte do processo de integração é atualizar e melhorar o conhecimento geral de domínio. Compreende as definições de tipo e o tesauro de mensuração.

As definições de tipo são revisadas baseadas no *feedback* de novos casos atualizados. Por exemplo, se o intervalo de valores do atributo "função do problema" foi definido como {engenheiro de qualidade, desenvolvedor de software, testador}, mas nos novos casos adquiridos está especificado como "gerente de projeto", a nova função após a revisão de sua relevância no ambiente específico é incluída no intervalo de valores possíveis.

Cada tipo é relacionado a um atributo específico no domínio da mensuração de software. Uma hierarquia de objetos relevantes é representada na Interface de Objeto (ver Figura 18).

Aqui, o engenheiro de conhecimento pode adicionar, modificar ou apagar objetos definidos na hierarquia.

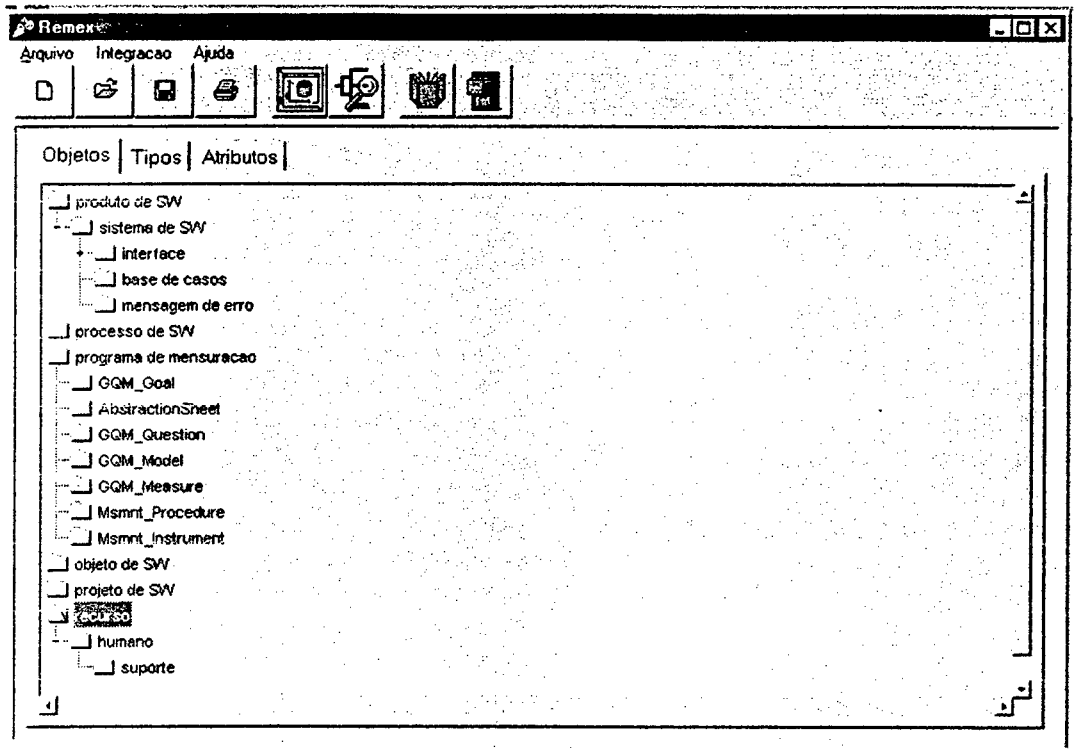


Figura 18 Exemplo de revisão de hierarquia de objetos

Em relação aos objetos, atributos relevantes podem ser definidos (ver Figura 19). Novamente, o engenheiro de conhecimento pode adicionar, modificar ou apagar atributos

relativos a objetos definidos na hierarquia de objetos.

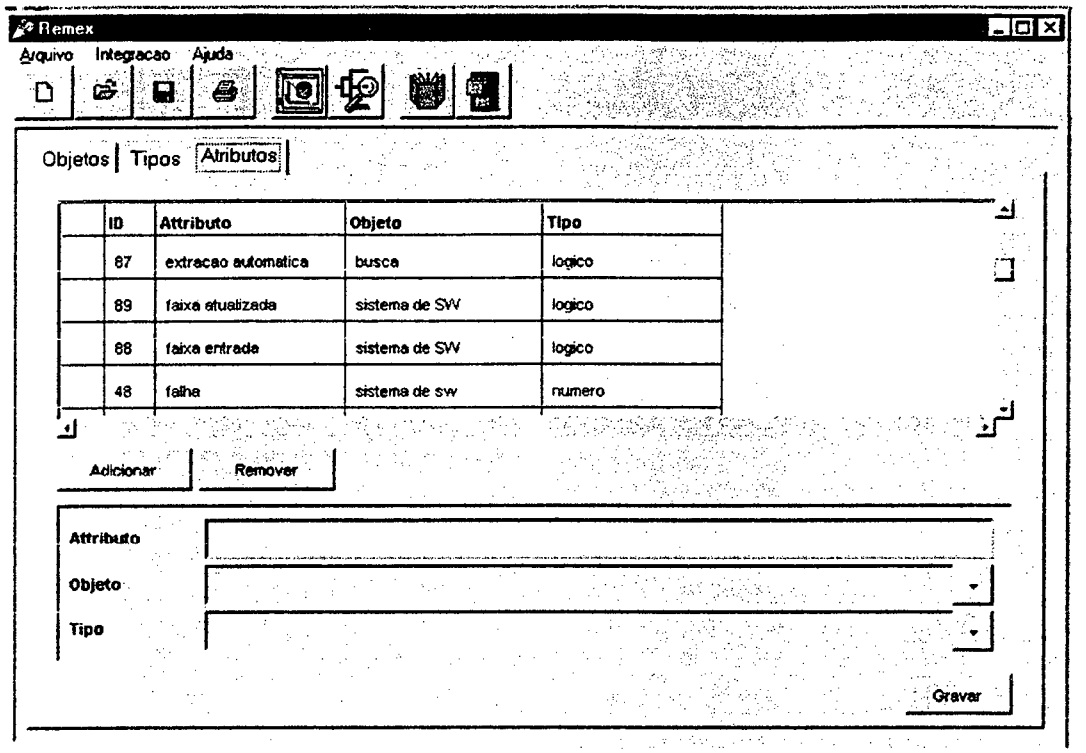


Figura 19 Exemplo da revisão de definições de atributos

Para cada atributo um tipo tem que ser definido. Aqui, cada um tipo básico pode ser selecionado ou um tipo específico pode ser definido pelo usuário no Gerenciador de Tipos. O Gerenciador de Tipos mostra a configuração de parâmetros das definições de tipo (ver Figura 21) e habilita a definição de tipos definidos pelo usuário. Para cada tipo os seguintes parâmetros são especificados:

- nome do tipo
- supertipo
- unidade (no caso de valores numéricos)
- intervalo de valores
- medida de similaridade local (especificando como o valor de similaridade local é calculado, ex., no caso de símbolos não ordenados pela tabela de similaridade)
- limiar local de similaridade definindo o limite de dois valores de um tipo a ser considerados similares

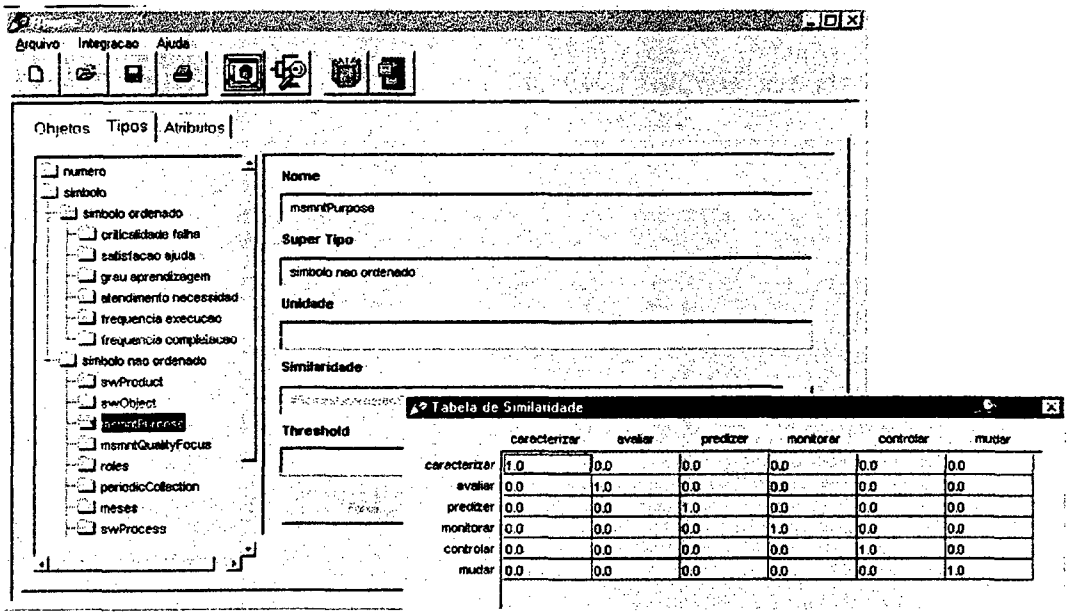


Figura 20 Exemplo de definição de tipo (tabela de similaridade)

Aqui, no exemplo dado, deseja-se adaptar o atributo função de uma organização existente. Então, nenhuma modificação ou melhoria na hierarquia de objeto, bem como definição de atributo, tem que ser feita. Como o novo valor é adicionado no intervalo de valores de tipo ("gerente de projeto"), a definição de tipo é modificada respectivamente.

A terminologia usada para a descrição de caso é revisada considerando o ambiente em particular.

Se quaisquer sinônimos são detectados na descrição do caso em relação aos casos armazenados na GQM-LL-KB, os termos e seus sinônimos são incluídos no tesauro de mensuração.

10.5.3. Melhoria da performance de recuperação

A contínua evolução e customização da GQM-LL-KB a um ambiente específico pode também necessitar a modificação da representação dos GQM-PSECs ou dos parâmetros do método de recuperação, incluindo o esquema de índices baseado no *feedback* da sua aplicação.

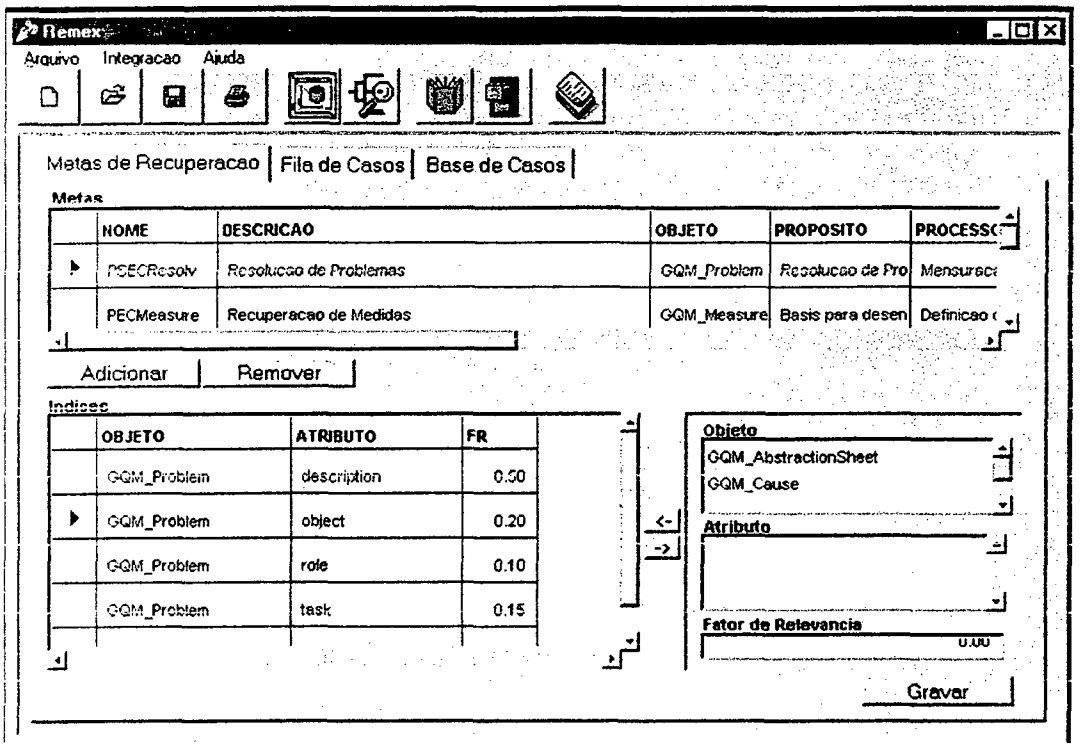


Figura 21 Exemplo de definição de parâmetros do método de recuperação

O Gerenciador de Recuperação da ferramenta GQM-LL facilita a adaptação do método de recuperação habilitando a configuração explícita dos parâmetro de recuperação (ver Figura 21).

O Gerenciador de Recuperação cobre principalmente o gerenciamento de todos parâmetros para o método de recuperação:

- nome
- meta de reutilização em termos de objeto, objetivo, ponto de vista do processo, contexto
- saída do processo de recuperação
- parâmetros α, β,γ,δ da medida de similaridade global
- limiar global
- lista de índices para o processo de recuperação baseada na estrutura de representação dos GQM-PSECs.

Qualquer modificação feita considerando os parâmetros recuperados resultam automaticamente na atualização do método de recuperação;

10.6 Discussão

A ferramenta GQM-LL integrada ao ambiente de mensuração REMEX foi experimentada como suporte substancial para o planejamento de mensuração de software. No contexto de um estudo empírico desenvolvido (ver Seção 11) a ferramenta GQM-LL foi usada pelos sujeitos do experimento fornecendo guia para o planejamento de um programa de mensuração.

A utilização do sistema é considerada pela maioria dos sujeitos do experimento sem qualquer problema ou pequenos problemas (ver Figura 22). Somente poucos sujeitos relataram problemas, que foram causados principalmente pela instalação e edição de problemas do REMEX do que pela utilização das funcionalidades de recuperação da ferramenta GQM-LL. A interface da ferramenta GQM-LL é considerada sempre ou na maioria das vezes compreensível pelos sujeitos (ver Figura 23).

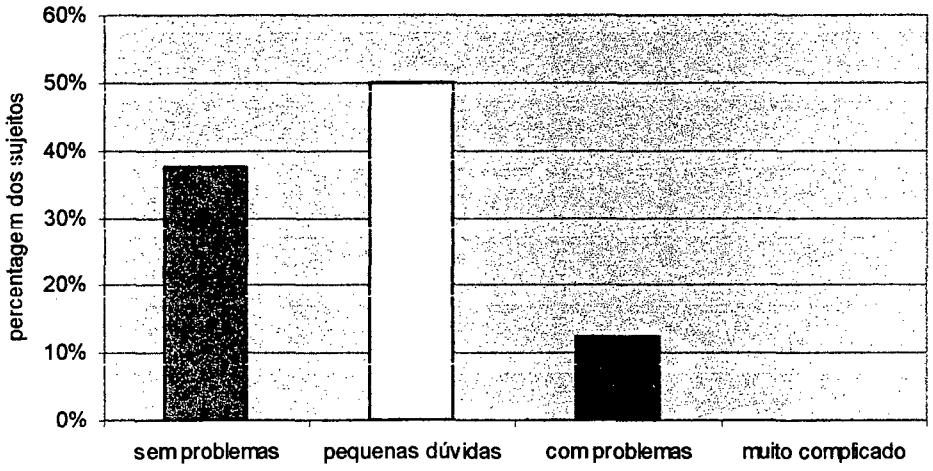


Figura 22 Utilização do sistema

Baseadas no *feedback* dos participantes do estudo, observações adicionais sobre a

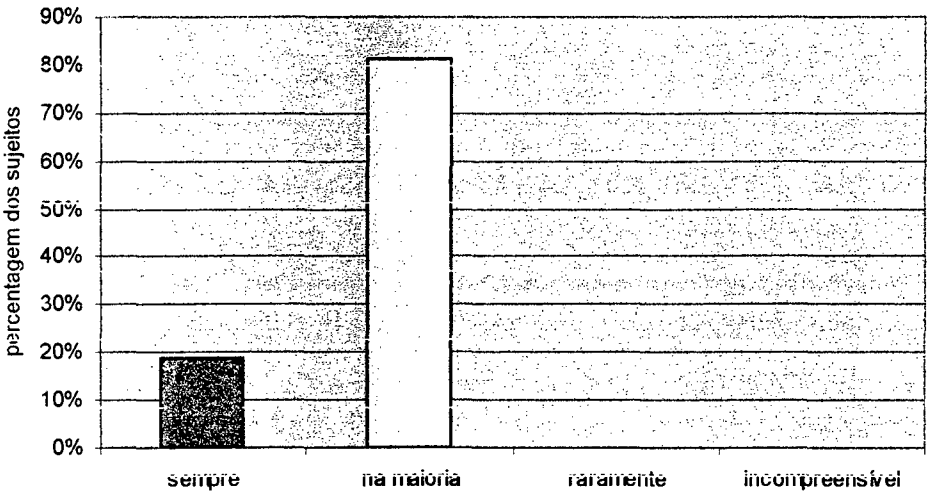


Figura 23 Compreensão do interface

ferramenta GQM-LL pode ser feita:

- a recuperação dos casos foi relatada como imediata
- os sujeitos confirmaram que os casos incluíam informações suficientes visando fazer decisões fundamentadas para a reutilização e,
- a forma de apresentação dos casos foi considerada adequada.

Esses dados e observações feitas pelos usuários através do estudo mostram que a representação de experiências em mensuração e a recuperação de GQM-PSECs é suportada adequadamente através da ferramenta GQM-LL que fornece um suporte fácil para compartilhar e comunicar lições aprendidas em mensuração.

11 Avaliação Empírica da Metodologia

Estudos empíricos na disciplina de Engenharia de Software são necessários [BSL99,BSH86]. Para realizar essa tarefa, estudos empíricos tem que ser desenvolvidos para auxiliar de melhor avaliar, predizer, entender, controlar e melhorar o processo de desenvolvimento e produto de software.

No contexto desta pesquisa que enfoca a reutilização sistemática de lições aprendidas em mensuração, a abordagem de desenvolvimento foi também avaliada através de um estudo empírico. A representação de experiências em mensuração em forma de GQM-PSECs e o processo de recuperação são analisados considerando sua adequação para compartilhar *know-how* em mensuração. Os benefícios da abordagem de planejamento de mensuração de software em comparação ao processo de planejamento sem reutilização de experiências em mensuração, relativos ao esforço e qualidade do programa de mensuração são investigados.

Esse capítulo descreve o experimento controlado desenvolvido, apresenta os resultados baseados na coleta de dados e as conclusões obtidas.

11.1 Visão geral

A hipótese principal nesse trabalho de pesquisa é que habilitar a reutilização sistemática de experiências em mensuração em forma de GQM-PSECs fornece suporte útil para o planejamento de programas de mensuração. Além disso, o método é assumido para melhorar a eficiência e efetividade ao planejamento de programas de mensuração GQM.

No contexto deste trabalho, um experimento controlado foi conduzido visando avaliar a hipótese declarada anteriormente. O experimento compara a tecnologia para o planejamento de programas de mensuração sem considerar a reutilização de lições aprendidas em mensuração (baseado em [BDR96,GHW95,BCR94b]), denotado como GQM-normal, com a tecnologia para planejamento de programas de mensuração GQM suportando sistematicamente a reutilização de lições aprendidas em forma de GQM-PSECs, denotados

como GQM-LL.

O experimento cobre os passos GQM2.2.2 - GQM3.1 do processo de planejamento de mensuração. O experimento foi integrado a um maior estudo empírico enfocando a avaliação da reutilização em geral no processo de planejamento de mensuração (ex., considerando também a reutilização de produtos GQM) [CG00] . Entretanto, como o foco desta pesquisa é particularmente na reutilização de lições aprendidas em mensuração, aqui discute-se somente aspectos relacionados a essa abordagem.

Sujeitos do experimento são estudantes da disciplina de mensuração de software do Curso de Pós-Graduação em Engenharia de Produção (PPGEP) da Universidade Federal de Santa Catarina (UFSC). Os 16 sujeitos que completaram o planejamento de mensuração, eram graduados nas ciências de engenharia (tanto em ciências da computação, engenharia elétrica ou engenharia de produção).

Foram 5 estudantes tempo integral e 11 sujeitos trabalhando 40 horas/semana em paralelo aos seus estudos. A experiência profissional variou de nenhuma (5 usuários) até 17 anos com uma média de 4 anos. Quase todos os sujeitos sem experiência tiveram pelo menos 1 ano de experiência em desenvolvimento de software em ambiente acadêmico (exceto um sujeito). Todos os sujeitos tiveram experiência em uso diário de software. Nenhum sujeito tinha planejado anteriormente um programa de mensuração seguindo a abordagem GQM, embora dois sujeitos relataram experiência em mensuração.

Para o experimento, os usuários foram divididos em dois grupos separados aleatoriamente. Um grupo, denotado como GQM-normal, usou o enfoque GQM para o planejamento de programa de mensuração sem suporte sistemático para a reutilização de lições aprendidas, enquanto que o outro grupo, denotado como GQM-LL, usou o enfoque GQM-R+ [Gre00], que suporta sistematicamente a reutilização de experiências em mensuração incluindo a reutilização de lições aprendidas. Cada sujeito planejou individualmente um programa de mensuração durante o experimento, desenvolvendo os produtos GQM descritos nos passos GQM2.2.2-GQM3.1 do processo GQM.

Todos os programas de mensuração planejados pelo sujeitos foram relacionados a

mesma meta de mensuração, que foi pré-definida para o estudo empírico. A meta de mensuração foi: Analisar o módulo de matrícula do software STELA (versão *totem*) para o propósito de melhoria com o foco de qualidade usabilidade sob o ponto de vista do usuário do Curso de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina.

O sistema STELA [Ste00] é um sistema de software usado para a matrícula, administração dos dados dos estudantes, etc. no Curso de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina. O sistema está disponível em *totems* no prédio da universidade, bem como através da Internet, mesmo o experimento enfocando a versão *totem* do sistema

O processo de planejamento de mensuração foi suportado pelo protótipo REMEX, que dá suporte para a edição de plano de programas de mensuração seguindo a abordagem GQM. A ferramenta GQM-LL foi integrada a ferramenta REMEX oferecendo facilidades para a reutilização de GQM-PSECs. Usuários do grupo GQM-normal acessaram somente as facilidades de edição, enquanto que os usuários do grupo GQM-LL, além das facilidades de edição da ferramenta REMEX, também acessaram as facilidades de reutilização da ferramenta GQM-LL. A GQM-LL-KB continha um total de 40 GQM-PSECs relatados em diferentes fases do processo de planejamento GQM. Material fornecido aos usuários continham material para leitura consistindo de artigos, capítulos de livros e relatórios sobre mensuração e o processo GQM.

11.2 Planejamento do experimento

O objetivo do experimento era avaliar se a reutilização sistemática de GQM-PSECs fornece suporte útil para o planejamento de programas de mensuração em melhora a eficiência (em termos de redução de esforço) e efetividade do planejamento de programas de mensuração (em termos de qualidade do plano de mensuração resultante).

As seguintes hipóteses foram investigadas:

Hipótese 1 Sujeitos aceitam e consideram como útil a reutilização de lições aprendidas

durante o processo de planejamento.

O objetivo é avaliar se os sujeitos consideram a reutilização de GQM-PSECs como útil para o planejamento de programas de mensuração. Em adição, a usabilidade do suporte fornecido através da reutilização de GQM-PSECs é avaliada também considerando a forma de representação dos GQM-PSECs. Isto inclui a clareza, relevância e totalidade das informações representadas nos GQM-PSECs.

Hipótese 2 Planejamento de Mensuração baseado no processo GQM-R+ incluindo a reutilização de GQM-PSECs reduz o esforço de planejamento de programas de mensuração.

Hipótese 3 Planejamento de mensuração baseado no processo GQM-R+ incluindo reutilização de GQM-PSECs aumenta a qualidade dos produtos GQM contidos no plano de mensuração.

A qualidade dos planos de programas de mensuração é avaliada em termos de boa definição dos produtos GQM desenvolvidos.

11.2.1. Programas de mensuração

Nesta seção fornece-se uma visão geral das metas de mensuração definidas e as respectivas questões analisadas para cada uma das hipóteses declaradas anteriormente. Mais informações sobre medidas, procedimentos de coleta de dados e instrumentos podem ser encontrados em [CG00].

Meta de mensuração 1. Analisar o enfoque GQM considerando sua aceitação e usabilidade sob o ponto de vista do pesquisador no contexto do experimento desenvolvido na PPGEP/UFSC.

Questões

Q1. Qual o grau de suporte fornecido através da reutilização de GQM-PSECs?

Q2. GQM-PSECs incluem informação suficiente para decidir se a solução do problema pode ser reutilizado ou não para resolver o problema atual?

Q3. GQM-PSECs são considerados como um recurso claro e compreensível?

Q4. A informação representada nos GQM-PSECs é relevante?

Meta de mensuração 2. Comparar o planejamento de mensuração baseada em GQM-normal e GQM-R+ (incluindo a reutilização de GQM-PSECs) enfocando o esforço sob o ponto de vista do pesquisador no contexto do experimento desenvolvido na PPGEP/UFSC.

Questão

Q5. Qual a média total de esforço gasto no planejamento de mensuração comparando os grupos (GQM-normal e GQM-LL) ?

Meta de mensuração 3. Comparar a planejamento de mensuração baseada em GQM usando GQM-normal e GQM-R+ (incluindo a reutilização de GQM PSECs) enfocando a percentagem de produtos GQM bem definidos contidos no plano do programa de mensuração sob o ponto de vista do pesquisador no contexto do experimento desenvolvido na PPGEP/ UFSC.

Questão

Q6. Qual a percentagem média de produtos GQM bem definidos (incluindo questões GQM, modelos de qualidade, medidas GQM, procedimentos de mensuração) comparando os dois grupos (GQM-normal e GQM-LL)?

11.2.2. *Design* experimental

O experimento destaca um fator: reutilizar GQM-PSECs. Visando minimizar efeitos, possíveis efeitos são igualmente distribuídos em todas as possíveis condições examinadas. O objetivo do grupos que foi alocado com sujeitos distribuídos de forma relativamente homogênea considerando possíveis fatores, tais como experiência em mensuração, disponibilidade de tempo e experiência em software. Os sujeitos foram distribuídos aleatoriamente nos grupos GQM-normal (sujeitos 1,2,3,4,5,6,7,8) ou GQM-LL (sujeitos 9,10,11,12,13,14,15,16).

11.2.3. Ameaças a validade

Um problema que pode ameaçar a validade dos resultados é o intervalo limitado da aplicação da abordagem de mensuração estudado. No contexto do experimento, todos os sujeitos planejaram programas de mensuração para uma meta de mensuração pré-definida visando eliminar qualquer efeito devido a diferentes metas de mensuração. A limitação complica a generalização dos resultados para a aplicação de abordagens de mensuração em geral dando enfoque a diferentes metas.

Outro problema envolve o pequeno número e escolha dos sujeitos do experimento. A maioria dos sujeitos não tinha experiência em mensuração, que faz com que a generalização dos resultados para engenheiros profissionais de garantia de qualidade seja difícil. Entretanto, como a reutilização de experiências em mensuração é especialmente indicada pelo pessoal de garantia de qualidade sem experiência, principiantes são uma população apropriada para estudar tais benefícios.

Devido a organização do experimento ter durado um período de 6 semanas e o fato de que o trabalho ter sido feito em maioria em casa, pouco controle foi possível considerando o intercâmbio entre os participantes e o uso de material adicional. Embora não tenha sido permitido no experimento, podem ter sido reutilizados sem planejamento, produtos de mensuração de colegas ou de literatura.

O relativo pequeno número de participantes (16) também complica a derivação de resultados estatísticamente relevantes.

11.2.4. Condução do experimento

O experimento foi organizado como exercício da disciplina de Mensuração de Software no Curso de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina durante o terceiro trimestre de 1999. Os sujeitos foram divididos aleatoriamente em dois grupos, GQM-normal ou GQMQ-LL. O número de sujeitos em cada grupo foi balanceado. O experimento começou com 20 participantes. Entretanto, dados relativos aos sujeitos que não completaram a tarefa foram desconsiderados, resultando em um total de 16

participantes (8 participantes em cada grupo).

O objetivo do exercício foi desenvolver um programa de mensuração para uma meta de mensuração pré-definida. A meta foi «Analisar o módulo de matrícula do sistema STELA (versão *totem*) considerando a usabilidade sob o ponto de vista de estudantes do curso de Pós-Graduação em Engenharia de Produção na Universidade Federal de Santa Catarina». O verdadeiro objetivo por trás do experimento, a comparação das abordagens GQM-normal e GQM-LL, foi explicada ao sujeitos.

Durante a disciplina, os sujeitos foram treinados passo-a-passo na aplicação do enfoque GQM. Cada fase do processo GQM foi explicado em detalhes, bem como, os respectivos produtos a serem desenvolvidos. Um exemplo completo de um programa de mensuração considerando uma meta de mensuração diferente da meta do experimento foi apresentado. Ambos os grupos seguiram o mesmo enfoque básico, incluindo as fases GQM2.2.2 - GQM3.1 como descrito em Seção 5. A cada semana, um passo era apresentado e ensinado durante a aula. Após disso, os sujeitos individualmente executaram as respectivas atividades GQM e desenvolveram os respectivos produtos GQM da etapa em particular.

A cada semana. os produtos relatados a fase específica GQM foram coletados, bem como, dados adicionais (ver Tabela 5) por exemplo, sobre o esforço gasto pelos sujeitos na respectiva tarefa GQM. Os produtos GQM desenvolvidos pelos sujeitos foram revisados pelos pesquisadores diretamente após cada etapa visando prevenir que os planos de mensuração tenham sido feitos completamente errados como a maioria dos sujeitos não tinha experiência em mensuração. Entretanto, durante o experimento nenhum dos planos de programa de mensuração precisou de interferência dos pesquisadores.

A duração total do experimento foi de 6 semanas. Visando adequar o esforço gasto no experimento como exercício de uma disciplina do curso de Pós-Graduação, o tamanho dos programas de mensuração desenvolvidos tinha em média 10-12 fatores de folhas de abstração.

Durante todo o experimento suporte contínuo foi fornecido visando resolver todos os problemas encontrados pelos sujeitos sobre o enfoque GQM, sobre a ferramenta ou sobre a organização do experimento.

11.2.5. Material do experimento

Como material da disciplina os sujeitos receberam uma coleção de artigos, relatórios e capítulos de livros sobre mensuração de software e sobre a abordagem GQM incluindo também um exemplo completo de um plano de programa de mensuração baseado em [GR99,GHW95].

Além disso, os sujeitos receberam uma cópia da ferramenta GQM-LL integrada ao ambiente de mensuração REMEX. Duas versões diferentes da ferramenta de suporta foram preparadas. O grupo GQM-normal recebeu uma versão incluindo somente facilidades de edição para o desenvolvimento do plano do programa de mensuração como parte da ferramenta REMEX.

O grupo GQM-LL recebeu uma versão que além de fornecer as facilidades de edição também incluía as facilidades de reutilização da ferramenta GQM-LL, permitindo o acesso a 40 GQM-PSECs armazenados na GQM-LL-KB.

Instrução a organização e execução do experimento, material da disciplina, a ferramenta REMEX e a ferramenta GQM-LL estavam disponíveis aos sujeitos no início do experimento. Os sujeitos também foram treinados para o uso das ferramentas. Uma visão geral de todas as funcionalidades da ferramenta foi dada no início do experimento e a cada semana as respectivas fases GQM foram explicadas em detalhes. As funcionalidades de reutilização da ferramenta GQM-LL foram apresentadas somente para o grupo GQM-LL

11.2.6. Instrumentos de coleta de dados

Além dos produtos GQM desenvolvidos pelos sujeitos, questionários foram usados com objetivo de coleta de dados adicionais de acordo com os procedimentos de mensuração definidos. Inclui os seguintes questionários:

- Questionário de caracterização dos sujeitos. O questionário coleta dados relevantes sobre as características de cada participante no início do experimento.
- Questionário de relatório da tarefa. Um questionário para cada tarefa do processo de planejamento GQM foi desenvolvido, o qual coleta dados relevantes sobre a performance da

tarefa em particular.

- Questionário de *feedback* final. O questionário coleta dados adicionais sobre as características dos sujeitos e *feedback* sobre a performance do processo de planejamento baseado em GQM.
- Guia de avaliação. Para guiar a avaliação dos planos de programa de mensuração desenvolvidos pelos sujeitos um guia de avaliação foi desenvolvido.

Para informações mais detalhadas sobre os questionários ver [CG00].

11.2.7. Procedimento de coleta e validação de dados

Os dados de mensuração necessários foram coletados através dos questionários pelos sujeitos e baseados na análise subjetiva de produtos GQM desenvolvidos pelos sujeitos.

Os sujeitos forneceram informações sobre suas características no início do experimento (Questionário de caracterização dos sujeitos). No final de cada tarefa GQM, os sujeitos completaram o respectivo Questionário de relatório da tarefa fornecendo dados sobre a execução e sobre problemas encontrados em uma determinada etapa. No fim do experimento, *feedback* foi fornecidos pelos sujeitos através do Questionário de *feedback* final.

Além disso, os produtos desenvolvidos pelos sujeitos foram analisados e avaliados pelos pesquisadores e dados sobre o número de produtos GQM e a boa definição foram determinados subjetivamente.

11.3 Resultados

Os dados coletados durante o experimento são analisados e interpretados em relação a hipótese declarada na seção 11.2.

Hipótese 1 Os sujeitos aceitam e consideram como útil a reutilização de lições aprendidas em mensuração durante o processo de planejamento

A Figura 24 apresenta os dados coletados dos sujeitos do grupo GQM-LL par avaliar a

aceitação de suporte fornecido através da reutilização sistemática de experiências em mensuração.

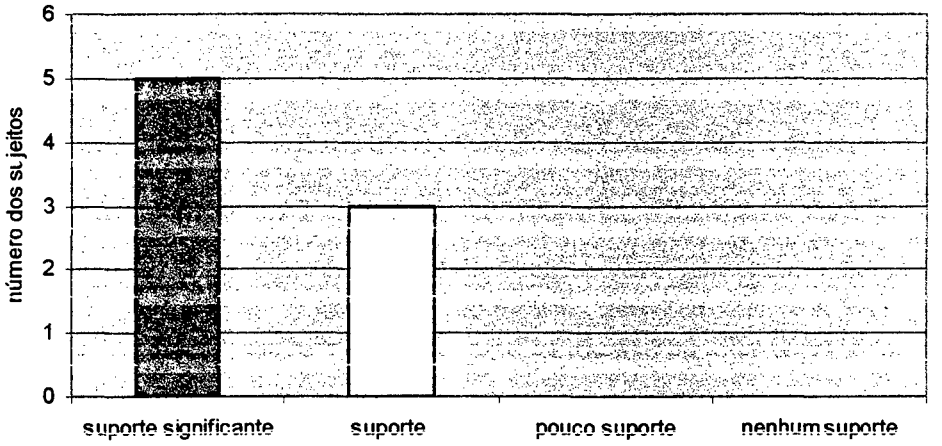


Figura 24 Dados subjetivos sobre o suporte fornecido

Todos os sujeitos consideraram que a possibilidade da reutilização e da recuperação baseada em similaridade de experiências como um suporte ao planejamento de programas de mensuração. Suas avaliações variaram de suporte a suporte significante em todas as tarefas do processo de planejamento consideradas no experimento. Nenhum dos sujeitos considerou as facilidades da reutilização como inúteis.

No caso da questão, se os GQM-PSECs são considerados um recurso claro e compreensível 100% dos sujeitos do grupo GQM-LL respondeu sim. Além disso, 100% dos sujeitos também considerou os GQM-PSECs completos e contendo somente informações relevantes.

Como resultado, pode-se concluir que o enfoque da reutilização de GQM-PSECs é apropriado e permite a comunicação e compartilhamento desse tipo de conhecimento visando apoiar o planejamento de programas de mensuração de software.

Entretanto, os sujeitos consideraram um melhor suporte possível se um maior número de casos pudesse estar disponível para reuso. Mas embora necessitando de mais casos, os sujeitos

consideraram o suporte fornecido como sendo útil. Além disso, através da contínua aquisição de novos casos integrada ao enfoque, espera-se que o suporte fornecido em um ambiente industrial aumente com o número de casos capturados.

Hipótese 2 O planejamento de mensuração baseado no processo GQM-R+ incluindo reutilização de GQM-PSECs reduz esforço no planejamento de programas de mensuração.

A Figura 25 na pagina 148, mostra graficamente os dados plotados representando o total de esforço gasto durante o planejamento de programa de mensuração (incluindo GQM2.2.2 - GQM3.1). Visando confirmar o efeito do uso do enfoque GQM-R+ (incluindo a reutilização de GQM-PSECs) sobre o esforço gasto no planejamento de mensuração, analisa-se os dados usando teste de estatística não paramétrica como não pode-se assumir uma distribuição normal dos dados e devido ao pequeno tamanho da amostra [FP97,LF91]. Aqui usa-se o teste Mann-Whitney, já que se compara dois grupos independentes [LF91] considerando as seguintes hipóteses:

Hipótese nula: O total de esforço em planejamento de mensuração (GQM 2.2.2 - GQM 3.1) não difere na aplicação de GQM-normal e GQM-LL.

Hipótese de pesquisa: O esforço total de planejamento de mensuração (GQM 2.2.2 - GQM 3.1) difere entre a aplicação de GQM-normal e GQM-R+.

O resultado do teste Mann-Whitney U é U=11. Conseqüentemente pode-se rejeitar a hipótese nula sobre o nível de significância de 0.05. Então, a diferença de esforço á julgada ser significante, apoiando a hipótese que sujeitos usando o enfoque GQM-R+ gastam menos esforço que sujeitos usando a abordagem sem a reutilização de conhecimento em mensuração.

Hipótese 3 O planejamento de mensuração baseado no processo GQM-R+ incluindo a reutilização de GQM-PSECs aumenta a qualidade dos produtos GQM contidos nos planos de programa de mensuração.

Visando ser efetivos, os planos de programa de mensuração tem que ser bem definidos. As questões GQM, modelos de qualidade, medidas GQM e procedimento de mensuração definidos nos planos de programas de mensuração foram subjetivamente classificados pelos pesquisadores como bem-definidos ou mal-definidos tomando em consideração se a terminologia usada foi definida e se todas as informações necessárias foram declaradas explicitamente e da maneira correta.

A Figura 26 na pagina 149 mostra os dados brutos da percentagem de produtos GQM mal-definidos no plano do programa de mensuração (incluindo questões GQM, modelos de qualidade, medidas GQM e procedimentos de mensuração) em relação ao número total dos produtos de mensuração desenvolvidos pelos sujeitos. Visando confirmar o efeito de fatores considerados no plano de programa de mensuração, analisa-se os dados usando o teste Mann-Whitney U para as seguintes hipóteses.

Hipótese nula: A percentagem de produtos GQM mal-definidos não difere entre se aplicar GQM-normal e GQM-R+.

Hipótese de pesquisa: A percentagem de produtos GQM mal-definidos difere entre se aplicar GQM-normal e GQM-R+.

O resultado do teste Mann-Whitney U é U=3, e conseqüentemente a hipótese nula pode ser rejeitada com significância 0.5.

Isso significa que a diferença em percentagem de produtos GQM mal-definidos é julgada ser relevante, suportando a hipótese que os planos de programas de mensuração desenvolvidos pelos sujeitos usando GQM-R+ incluem um percentagem maior de produtos GQM bem-definidos que aqueles desenvolvidos usando GQM-normal.

11.4 Discussão

Os dados coletados no experimento apoiaram os hipóteses, indicando que a abordagem GQM-LL fornece suporte útil para o planejamento de programas de mensuração GQM em comparação ao enfoque GQM sem considerar a reutilização sistemática de conhecimento em

mensuração. Os GQM-PSECs tem mostrado ser uma forma apropriada de compartilhar e comunicar experiências em mensuração guiando especialmente pessoas sem experiência no planejamento de programas de mensuração de software. O suporte foi considerado útil apesar do pequeno número de GQM-PSECs disponíveis na GQM-LL-KB (uma situação realística quando se inicia o estabelecimento de uma GQM-LL-KB na prática). Além disso, a representação e recuperação de experiências em forma de GQM-PSECs foi considerada apropriada e útil pelos participantes do experimento.

Entretanto, visando aumentar a quantidade de resultados confiáveis do experimento, replicações do estudo precisam ser feitas. Pela execução das replicações certas ameaças a validade podem ser apontadas: validade interna, que define o grau de confiabilidade em relacionamentos de causa-efeito entre fatores de interesse e os resultados observados; e a validade externa, que define a extensão na qual as conclusões no contexto experimental podem ser generalizadas a um contexto especificado na hipótese de pesquisa [BSL99].

Visando avaliar o uso da tecnologia em ambientes industriais e confirmar sua efetividade na prática industrial, estudos empíricos tem que ser feitos também em ambientes industriais, por exemplo, em forma de estudos de casos [FP97]. Embora estudos de casos sejam menos controlados que um experimento controlado, eles podem ser usado para entender o que está acontecendo em uma determinada organização. Esses estudos fornecem *feedback* importante em usabilidade e efetividade da tecnologia e no modo como deve ser introduzido no futuro para obter maior aceitação.

12 Conclusão

Nesta pesquisa, foi analisado o potencial para melhoria do processo de planejamento de mensuração de software através do suporte sistemático para a reutilização de experiências em mensuração. O foco do trabalho é operacionalizar o compartilhamento e comunicação de lições aprendidas em mensuração na organização, promovendo a criação de um corpo organizacional de *know-how* em mensuração e a aplicação amplamente distribuída de mensuração em projetos industriais

Baseados nos requisitos identificados, foram avaliados vários enfoques para a reutilização de experiências no domínio da mensuração de software. Devido ao fato do Raciocínio Baseado em Casos ter mostrado ser uma abordagem ótima para a operacionalização da Fábrica de Experiência fornecendo um suporte amplo para o desenvolvimento de sistemas baseados em conhecimento, foi desenvolvida um metodologia para a operacionalização da reutilização de experiências em mensuração integrando e melhorando técnicas de Raciocínio Baseado em Casos. Baseada em um análise detalhada do processo de planejamento de mensuração baseada em GQM e na manipulação de conhecimento tácito através do enfoque RBC, uma representação compreensiva foi desenvolvida para a apresentação de lições aprendidas em mensuração. Um método inovador de recuperação baseado em RBC foi aplicado compreendendo recuperação orientada a metas e baseada em similaridade para lições aprendidas. Um método para adaptação do enfoque a um ambiente especifico, bem como a contínua manutenção da GQM-LL-KB é fornecida. Baseado na metodologia foi desenvolvido um protótipo de ferramenta para suportar a aplicação do enfoque na prática.

O desenvolvimento do enfoque GQM-LL e da ferramenta são integrados ao enfoque REMEX, uma abordagem compreensiva para reutilização de vários tipos de conhecimento com o objetivo de suportar o processo de planejamento de mensuração. Dentro do enfoque REMEX, a abordagem GQMQ-LL representa uma melhoria substancial pois permite uma manipulação sistemática e explícita de conhecimento tácito descrevendo como fazer mensuração e como adaptar o processo de planejamento a diferentes ambientes. A integração

da abordagem GQM-LL no ambiente de mensuração mostrou-se essencial para a aquisição de benefícios esperados pois facilita o acesso a conhecimento representativo por possíveis usuários.

Os resultados do estudo empírico, analisando a abordagem e a ferramenta GQM-LL mostra que a reutilização sistemática de experiências em mensuração em forma de GQM-PSECs fornece um suporte útil para o planejamento de programas de mensuração. Além disso, o efeito positivo do método no planejamento de programas de mensuração considerando a redução de esforços de planejamento e melhoria da qualidade de planos de programas de mensuração foi mostrado. Visando generalizar esses resultados outros estudos empíricos são necessários, por exemplo, em diferentes ambientes considerando diferentes metas de mensuração, etc. Espera-se, especialmente aplicações em ambientes industriais, trazer resultados de valor e a coleção de casos reais.

Devido a disponibilidade da ferramenta GQM-LL permitir a coleta contínua e sistemática de lições aprendidas em mensuração, esse conhecimento coletado pode ser usado como base para o refinamento futuro e melhoria o enfoque GQM. Guias gerais e heurísticas para a aplicação da mensuração na prática para diferentes tipos de ambientes podem ser derivadas habilitando o desenvolvimento sistemático de um corpo de conhecimento em mensuração de software.

A abordagem desenvolvida nesta pesquisa foi aplicada na área de mensuração de software. Ainda, a transferência da metodologia em outras áreas da Engenharia de Software com objetivo de dar suporte a reutilização de experiências, (ex., como fazer inspeções) é possível, mas requer uma revisão cuidadosa e adaptação da representação de casos, indexação e parâmetros da recuperação em um contexto particular.

Referências

[AAB+95] K.-D. Althoff, E. Auriol, R. Barletta, and M. Manago. A Review of Industrial Case-Based Reasoning Tools. AI Intelligence. Oxford (UK), 1995.

[ABG+98] K.-D. Althoff, A. Birk, C. Gresse von Wangenheim, C. Tautz. CBR for Experimental Software Engineering. In H. D. Burkhard et al. (eds.), Case-Based Reasoning Technology from Foundations to Applications, Springer Verlag, 1998.

[ABH+97] K.-D. Althoff, A. Birk, S. Hartkopf, C. Tautz. Packaging and Representing Experiences, IESE Report No. 022.97/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, 1997.

[ABT98] K.-D. Althoff, F. Bomarius, C. Tautz. Using Case-based Reasoning Technology to Build Learning Organizations. In Proceedings of the the Workshop on Organizational Memories at the European Conference on Artificial Intelligence '98, England, 1998.

[AK98] J. C. Abib, T. G. Kirner. GQM-Plan: Uma Ferramenta para Apoiar Avaliações de Qualidade de Software. IX Conferência Internacional de Tecnologia de Software: Qualidade de Software, Brazil, 1998.

[Alt97] K.-D. Althoff. Candidate Technologies for Supporting the Realization of an Experience Factory. Technical Report IESE, February 1997.

[Alt96] K.-D. Althoff. Evaluating Case-Based Reasoning Systems: The Inreca Case Study. Habilitation, University of Kaiserslautern, Germany, 1996.

[Alt89] K.-D. Althoff. Knowledge Acquisition in the Domain of CNC Machine Centers; the MOLTKE Approach, In Proc. of the third European Workshop on Knowledge-Based Systems, France, 1989..

[ANT99] K.-D. Althoff, M. Nick, C. Tautz. CBR-PEB: An Application Implementing Reuse Concepts of the Experience Factory for the Transfer of CBR System Know-How. Proc. of the 7th German Workshop on Case-Based Reasoning, Germany, 1999.

[AP94] A. Aamodt, E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. AI Communications, 17(1), 1994.

[Ash91] K. Ashley. Modeling Legal Arguments: Reasoning with Cases and Hypotheticals. MIT Press, Bradford Books, Cambridge, 1991.

[Aur86] Novo Dicionário Aurélio da Língua Portuguesa, 2a edição, Editora Nova Fronteira, 1986.

[AW97] K.-D. Althoff, W. Wilke. Potential Uses of Case-based Reasoning in Experienced Based Construction of sOftware Systems and Business Process Support. In R. Bergmann, W. Wilke (eds.), Proceedings of the 5th German Workshop on Case-Based Reasoning, LSA-97-01E, Centre for Learning Systems and Appli-

cations, University of Kaiserslautern, March 1997.

[BAB+87] B. A. Burton, R. W. Aragon, S. A. Bailey, K. D. Koehler, L. A. Mayes. The Reusable Software Library. IEEE Software, 4(7), July 1987.

[Bar88] R. Bareiss. Exemplar-based knowledge acquisition: A unified approach to concept representation, classification and learning, Ph.D. Dissertation, Technical Report AI88-83, University of Texas at Austin, Dep. of Computer Sciences 1988.

[Bas92] V. R. Basili. Software Modelling and Measurement: The Goal/Question/Metric Paradigm. Technical Report CS-TR-2956, Department of Computer Science, University of Maryland, College Park, MD 20742, September 1992.

[BC95] V. R. Basili, G. Caldiera. Improve Software Quality by Reusing Knowledge and Experience. Sloan Management Review, Fall 1995.

[BCM+92] V. R. Basili et al. The Software Engineering Laboratory - An Operational Software Experience Factory. Proceedings of the 14th International Conference on Software Engineering, 1992.

[BCR94a] V. R. Basili, G. Caldiera, H. D. Rombach. Experience Factory. In John J. Marciniak (ed.), Encyclopedia of Software Engineering, vol.1. John Wiley & Sons, 1994.

[BCR94b] V. R. Basili, G. Caldiera, H. D. Rombach. Goal/Question/Metric Approach. In J. Marciniak (ed.), Encyclopedia of Software Engineering, volume 1. John Wiley & Sons, 1994.

[BDR96] L. C. Briand, C. M. Differding, H. D. Rombach. Practical Guidelines for Measurement-based Process Improvement. *Software Process*, 2(4), December 1996.

[BDT96] A. Bröckers, C. Differding, G. Threin. The Role of Software Process Modeling in Planning Industrial Measurement Programs. Proceedings of the METRICS' 96, International Conference of Software Engineering, 1996.

[BE97] L. Briand, K. El Emam. Cost and Benefits of Software Process Improvement. Technical Report, Fraunhofer Institute for Experimental Software Engineering, 1997.

[BEM96] L. Briand, K. El Emam, S. Morasca. On the Application of Measurement Theory in Software Engineering. Journal on Empirical Software Engineering, 1 (1), 1996.

[BM96] J.M. Barr, R.V. Magaldi. Corporate Knowledge Management for the Millennium. In I. Smith, B. Faltings (eds.), Advances in Case-Based Reasoning, Springer Verlag, 1996.

[BR91] V. R. Basili, H. D. Rombach. Support for Comprehensive Reuse. IEEE Software Engineering Journal, 6(5), 1991.

[BR88] V. R. Basili, H. D. Rombach. The TAME Project: Towards Improvement-Ori-

ented Software Environments. IEEE Transactions on Software Engineering, SE-14(6), 1988.

[BSL99] V. Basili, F.Shull, F. Lanubile. Using Experiments to Build a Body of Knowledge, XX, 1999.

[BSH86] V. R. Basili, R.W. Selby, D. H. Hutchens. Experimentation in Software Engineering, IEEE Transactions on Software Engineering, 12(7), July 1986.

[BW96] R. Bergmann, W. Wilke. On the Role of Abstraction in Case-Based Reasoning. In Proceedings of European Workshop of Case-Based Reasoning, 1996.

[BW84] V. R. Basili, D. M. Weiss. A Methodology for Collecting Valid Software Engineering Data. IEEE Transactions on Software Engineering, SE-10(6):728-738, 1984.

[CEM96] The CEMP Consortium. Customized Establishment of Measurement Programs. Final Report, ESSI Project Nr. 10358, Germany,1996. (http://www.iese.fhg.de/Services/Projects/Public-Projects/Cemp.html)

[CFF+98] G. Cugola, P. Fusaro, A. Fuggetta, C. Gresse, L. Lavazza, S. Manca, M. R. Pagone, G. Ruhe, R. Soro. A Case Study of Evaluating Configuration Management Practices with Goal-Oriented Measurement. In Proceedings of the 4th Int. IEEE Symposium on Software Metrics, 1997.

[CG00] L. Correa, C. Gresse von Wangenheim. Planejamento e Execução de Avalição Experimental do Abordagem REMEX. Working Paper, Universidade Federal de Santa Catarina, in progress 2000.

[Cin97] D. Cindric. A Tool for Supporting the Development of Measurement Plans in GQM-Based Measurement Programs. Projektarbeit, Fraunhofer Institut for Experimental Software Engineering/Computer Science, University of Kaiserslautern, Germany,1997.

[CM96] J.R. Cho, R.C. Mathews. Interactions Between Mental Models Used in Categorization and Experiential Knowledge of Specific Cases. Journal of Experimental Psychology, 49A (3), 1996.

[Das92] M. K. Daskalantonakis. A Practical View of Software Measurement and Implementation Experiences within Motorola. Transactions on Software Engineering, 18(11), November 1992.

[DHL95] C. Differding, B. Hoisl, C. Lott. Technology Package for the Goal/Question Metric Paradigm. Technical Report 281-96, Department of Computer Science, University of Kaiserslautern, Germany, 1995.

[Dif93] C. Differding. An Object Model for Supporting the GQM Paradigm (in German). Master Thesis, Department of Computer Science, University of Kaiserslautern, Germany, June 1993.

[DR96] C. Differding, H. D. Rombach. Kontinuierliche Software-Qualitätsverbesserung in der industriellen Praxis. In R. Dumpke, C. Ebert (eds.), Software-Metriken in der Praxis, Springer Verlag, 1996.

[EB97] K. El Eman,L. Briand. Costs and Benefits of Software Process Improvement. Technical Report ISERN-97-12, ISERN Network, 1997.

[FGG+96] C. Fernández-Chamizo, P. A. González-Calero, M. Gómez-Albarrán, L. Hernández-Yáñez. Supporting Object Reuse Through Case-Based Reasoning. In I. Smith, B. Faltings (eds.), Advances in Case-Based Reasoning, Lectures Notes in Artificial Intelligence 1168, Springer Verlag, 1996.

[FLM+98] A. Fuggetta, L. Lavazza, S. Morasca, S. Cinti, G. Oldano, E. Orazi. Applying GQM in an Industrial Software Factory. ACM Transactions on Software Engineering and Methodology, October 1998.

[FN87] W. B. Frakes B.A. Nejmeh. An Information System for Software Reuse. In Proceedings of the Tenth Minnowbrook Workshop on Software Reuse, 1987.

[FP97] N.E. Fenton, S. L. Pfleeger. Software Metrics - A Rigorous & Practical Approach. Thompson Computer Press, 1997.

[Fri94] R. Friedl. A Tool for Evaluating GQM Plans (in German). Master Thesis, Department of Computer Science, University of Kaiserslautern, Germany, June 1994.

[FS84] G. Fischer, M. Schneider. Knowledge-Based Communication Processes in Software Engineering. In Proceedings of the Seventh International Conference of Software Engineering, Florida, 1984.

[FWD97] G. R. Finnie, G. W. Wittig, J.-M. Desharnais. Estimating Software Development Effort with Case-Based Reasoning. In Proceedings of the 2nd Int. Conference on Case-Based Reasoning, Providence, RI, July 1997.

[GAB99] C. Gresse von Wangenheim, K.-D. Althoff, R. M. Barcia. Intelligent Retrieval of Software Engineering Experienceware. In Proceedings of the 11th Int. Conference on Software Engineering and Knowledge Engineering, Germany, 1999.

[GAB+98] C. Gresse von Wangenheim, K.-D. Althoff, R. M. Barcia, C. Tautz. Evaluation of Technologies for Packaging and Reusing Software Engineering Experiences. IESE-Report No. 055.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern (Germany), 1998.

[GB98] C. Gresse, L. C. Briand. Requirements for the Knowledge-Based Support of Software Engineering Measurement Plans. Journal of Knowledge-Based Systems, 11, November 1998.

[GB97] C. Gresse, L.C. Briand. Requirements for the Knowledge-Based Support of Software Engineering Measurement Plans. In Proceedings of the 9th Int. Conference on Software Engineering and Knowledge Engineering, Spain, 1997.

[GF97] P. A. Gonzáles, C. Fernández. A Knowledge-based Approach to Support Software Reuse in Object-oriented Libraries. In Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering, Madrid, Spain, June 1997.

[GHR+97] C. Gresse, B. Hoisl, H. D. Rombach, G. Ruhe. Kosten-Nutzen-Analyse von GQM-basiertem Messen und Bewerten: Eine replizierte Fallstudie. In O. Grün and L. J. Heinrich (eds.), Wirtschaftinformatik - Ergebnisse empirischer Forschung, Springer Verlag, 1997.

[GHW95] C. Gresse, B. Hoisl, J. Wüst. A Process Model for GQM- Based Measurement. Technical Report STTI-95-04-E, Software Technology Transfer Initiative, University of Kaiserslautern, Germany, 1995.

[Gib94] W.W. Gibbs. Software's Chronic Crisis. Scientific American, November 1994.

[GMA+98] C. Gresse von Wangenheim, A. R. Moraes, K. D. Althoff, R. M. Barcia, R. Weber, A. Martins. Case-Based Reasoning Approach to Reuse of Experiential Knowledge in Software Measurement Programs. In Proceedings of the 6th German Workshop on Case-Based Reasoning, Germany, 1998.

[Gom97] H. Gómez. A Cognitive Approach to Software Reuse Applying Case-based Reasoning to Mutant Cases. In Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering, Madrid, Spain, June 1997.

[GR99] C. Gresse, G. Ruhe. Análise de Custo e Beneficio de Mensuração Baseada em GQM. Proc. of X Conferência International de Tecnologia de Software: Qualidade de Software, Brazil, 1999.

[Gre00] C. Gresse von Wangenheim. GQM-R+ - A Process Model for Reused-Based Planning of GQM-Measurement Programs. Technical Report GeNESS002.00E, Centro GeNESS, Federal University of Santa Catarina, 2000.

[Gre99] C. Gresse von Wangenheim. REMEX - A Case Based Approach for Reuse of Software Measurement Experienceware. In Proceedings of 3rd Int. Conference on Case-Based Reasoning, Germany, 1999.

[Gre98] C. Gresse von Wangenheim. Knowledge Management in Experimental Software Engineering - Create, Renew, Build and Organize Knowledge Assets. Position Paper. In Proceedings of the 10th Int. Conference on Software Engineering and Knowledge Engineering, San Francisco, 1998.

[Gre97] C. Gresse von Wangenheim. Qualidade de Software - Melhoramento baseado em Mensuração. In Proceedings of I Semana Tecnologica da UNIVALI, Universidade do Vale do Itajai, Florianópolis, Brazil, 1997.

[GRR98] C. Gresse von Wangenheim, H. D. Rombach and G. Ruhe. Tutorial on Melhoramento de Software Baseado em Mensuração - Como Aplicar GQM na Pratica?. Proceedings of the IX CITS - Conferência Internacional de Tecnologia de Software: Qualidade de Software, Curitiba, Brazil, 1998.

[GRR96] C. Gresse, H. D. Rombach, G. Ruhe. Tutorial on A Practical Approach for Building GQM-Based Measurement Programs - Lessons Learned from Three Industrial Case Studies. Proceedings of the X SBES - 10° Simpósio Brasileiro de Engenharia de Software, São Carlos, Brazil, 1996.

[GRR94] H. Günther, H. D. Rombach, G. Ruhe. Kontinuierliche Qualitätsverbesserung in der Software Entwicklung - Erfahrungen bei der Allianz Lebensversicherungs-AG (in German). Wirtschaftsinformatik 38, 1994.

[GRW+99] C. Gresse von Wangenheim, M. R. Rodrigues, A. von Wangenheim, R. M. Barcia. O Uso de Fábricas de Experiência em Software Engineering. Developers' Magazine, Nov. 1999

[GU97] C. Gresse von Wangenheim, J. M. Umbelino. Melhoramento de Qualidade de Software no CIASC. Seminário de Gestão da Qualidade Total, Brazil, December 1997.

[GW00] C. Gresse von Wangenheim, A. von Wangenheim. Introducing Effective MEasurement-Based Effort and Schedule Management in Small Software Companies. Technical Report Centro GeNESS, Universidade Federal de Santa Catarina, 2000

[GWB98] C. Gresse von Wangenheim, A. von Wangenheim, R. M. Barcia. Case-Based Reuse of Software Engineering Measurement Plans. In Proceedings of the 10th Int. Conference on Software Engineering and Knowledge Engineering, San Francisco, 1998.

[GT99] C. Gresse von Wangenheim, C. Tautz (eds.). Proceedings of the Workshop on "Practical Case-Based Reasoning Strategies for Developing Learning Organizations" at the 3. International Conference on Case-Based Reasoning, Germany, 1999.

[Hen97] S. Henninger. Capturing and Formalizing Best Practices in a Software Development Organization. In Proceedings of the 9th Int. Conference on Software Engineering & Knowledge Engineering, Spain, June 1997.

[Hen95] S. Henninger. Information Access Tools for Software Reuse, Journal of Systems and Software, 30(3), 1995.

[HK97] F. Houdek, H. Kempter. Quality Patterns - An Approach to Packing Software Engineering Experience. Software Engineering Notes, 22(3), 1997.

[HOR+96] B. Hoisl, M. Oivo, G. Ruhe, F. van Latum. No Improvement without Feedback: Experiences from Goal-oriented Measurement at Schlumberger. Fifth European Workshop on Software Process Technology, France, October 1996.

[IK96] T. Isakowitz, R. J. Kauffman. Supporting Search for Reusable Software Objects. IEEE Transactions on Software Engineering, 22(6), June 1996.

[Jal97] P. Jalote. An Integrated Approach to Software Engineering. Springer Verlag, 1997.

[KL96] H. Kempter, F. Leippert. Systematische Software-Qualitätsverbesserung durch zielorientiertes Messen und Bewerten sowie explizite Wiederverwendung des Software-Entwicklungs-Know-how (SoftQuali). Proceedings of the BMBF-Seminar Software Technology, Berlin, March 1996.

[Kol93] J. L. Kolodner. Case-Based Reasoning. Morgan Kaufmann, San Francisco, Cal-

ifornia, 1993.

[Kol83a] J. Kolodner. Maintaining Organization in a Dynamic Long-term Memory. Cognitive Science, Vol 7, 1983.

[Kol83b] J. Kolodner.Reconstructive Memory, a Computer Model, Cognitive Science, Vol. 7, 1983.

[Kot89] P. Koton. Using Experience in Learning and Problem Solving. Ph.D. Thesis, Technical Report MIT/LCS/TR-441, Massachusetts Institute of Technology, Laboratory of Computer Science, 1989.

[Kra97] K. Kramer. A Tool for Supporting the Development of Data Collection Instruments in GQM-based Measurement Programs. Projektarbeit, Fraunhofer Institut for Experimental Software Engineering/Computer Science, University of Kaiserslautern, Germany,1997.

[KS96] H. Kitano, H. Shimazu. The Experience-Sharing Architecture: A Case Study in Corporate-Wide Case-Based Software Quality Control. D. Leake (ed.), Case-Based Reasoning,: Experiences: Lessons Learned & Future Directions, 1996.

[LF91] J. Levin, J. A. Fox. Elementary Statistics in Social Research. Harper Collins, 1991.

[LHI97] B. Lees, M. Hamza, C. Irgens. Applying Case-Based Reasoning to Software Quality Management. In Proceedings of the 2. International Conference on Case-Based Reasoning, 1997.

[LJ88] L. Latour, E. Johnson. SEER: A Graphical Retrieval System for Reusable Ada Software Modules. In Proceedings of the 3rd Int. Conference on Ada Applications and Environments, IEEE Computer Society Press, 1988.

[LS98] W. Lam, V. Shankararaman. Managing Change During Software Development: An Incremental, Knowledge-Based Approach. In Proceedings of 10th Int. Conference on Software Engineering and Knowledge Engineering, San Francisco Bay, California, 1998.

[LSO+98] F. van Latum, R. van Solingen, M. Oivo, B. Hoisl, H. D. Rombach, G. Ruhe. Adopting GQM-Based Measurement in an Industrial Environment. IEEE Software, 15(1), January/Febuary 1998.

[Ma95] M. van Maris. GQM-DIVA - A Tool for Defining, Interpreting, and Validating GQM Plans (in German). Master Thesis, Department of Computer Science, University of Kaiserslautern, Germany, June 1995.

[Ma93] M. van Maris. A Syntax Oriented Editor for GQM Plans (in German). Projektarbeit, Department of Computer Science, University of Kaiserslautern, Germany, June 1993.

[MBC+94] M. Manago, R. Bergmann, N. Conruyt, R. Traphoener, F. Maurer, S. Wess, K.-D. Althoff, S. Dumont. Casuel: A Common Case Representation Language. Technical Report Deliverable D1, Version 2.01, Esprit Project Inreca (P6322), 1994.

[MBK91] Y. S. Maarek, D. M. Berry, G. E. Kaiser. An Information Retrieval Approach for Automatically Constructing Software Libraries. IEEE Transactions on Software Engineering, 17(8), August 1991.

[MVP92] T. Mukhopadhyay, S. S. Vicinanza, M. J. Prietula. Examining the Feasability of a Case-Based Reasoning Model for Software Effort Estimation. MIS Quarterly, 16(2), 1992.

[NAS94] National Aeronautics and Space Administration. Software Measurement Guidebook. Technical Report SEL-84-101, NASA Goddard Space Flight Center, Greenbelt MD 20771, July 1994.

[Nie90] J. Nielsen. Navigating Through Hypertext. Communications of the ACM, 33(3), March 1990.

[Nor93] D.A. Norman. Things that make us smart: Defending human attributes in the age of the machine. Addison-Wesley Reading, 1993.

[NT95] I. Nonaka, T. Takeuchi. The Knowledge-Creating Company. Oxford University Press, Cambridge, UK, 1995.

[OB92] M. Oivo, V.R. Basili. Representing Software Engineering Models: The TAME Goal Oriented Approach. IEEE Transactions on Software Engineering, 18(10), October 1992.

[OHP+92] E. Ostertag, J. Hendler, R. Prieto-Díaz, C. Braun. Computing Similarity in a Reuse Library System: An AI-Based Approach. ACM Transactions on Software Engineering and Methodology, 1(3), July 1992.

[Oiv94] M. Oivo. Quantitative Management of Software Production Using Object-Oriented Models. Dissertation. VTT Electronics, Oulu, Finland, 1994.

[PB86] B. Porter, R. Bareiss. PROTOS: An Experiment in Knowledge Acquisition for Heuristic Classification Tasks. In Proceedings of the First Int. Meeting on Advances in Lerning (IMAL), France, 1986.

[PER96] The PERFECT Consortium. GQMaspect. ESPRIT Project No. 9090 "PERFECT", Kaiserslautern, Germany, 1996. (http://www.iese.fhg.de/Services/Projects/Public-Projects/Perfect/Tools/GQMtools/asections.html)

[PG00] M. Pacheco, C. Gresse von Wangenheim. Metodologia para a Representação de Experiências baseado em Casos, Working paper, Universidade Federal de Santa Catarina, in progress 2000.

[PH90] C.K. Prahalad, G. Hamel. The Core Competence of the Corporation. Harvard Business Review, 68(3), May 1990.

[PJC+97] S. L. Pfleeger, R. Jeffrey, B. Curtis, B. Kitchenham. Status Report on Software Measurement. IEEE Software, 14(2), March 1197.

[Pol66] M. Polanyi. The Tacit Dimension. London: Routledge & Kegan Paul, 1966.

[Qui86] J.R. Quinlan. Induction of Decision Trees. Machine Learning, 1(1), 1986.

[Ric99] M.M. Richter. Tutorial on Case-Based Reasoning. Department of Computer Science, University of Kaiserslautern, Germany, 1999.

[Ric95] M.M. Richter. On the Notion of Similarity in Case-Based Reasoning. G. della Riccia et.al (eds.), Mathematical and Statistical Methods in Artificial Intelligence, Springer Verlag, 1995.

[RG00] M. R. Rodrigues, C. Gresse von Wangenheim. REMEX - A Reuse-Based Measurement Planning System.Working paper, Federal University of Santa Catarina, in progress.

[Rom91] H. D. Rombach. Practical Benefits of Goal-Oriented Measurement. Software Reliability and Metrics, Elsevier Applied Science, 1991.

[RS89] C.K. Riesbeck, R.C. Schank. Inside Case-Based Reasoning. Lawrence Erlbaum, 1989.

[Ruh98] G. Ruhe. Knowledge Management and Empirical Software Engineering. Position Paper. In Proceedings of the 10th Int. Conference on Software Engineering and Knowledge Engineering, San Francisco, California, 1998.

[RW91] M.M. Richter, S. Wess. Similarity, Uncertainty and Case-based Reasoning in PADTEX. In R.S. Boyer (ed.): Automated Reasoning - Essays in Honour of Woody Bledsoe, Kluwer, 1991.

[SLO95] R. van Solingen, F. van Latum, M. Oivo, E. Berghout. Application of Software Measurement at Schlumberger RPS: Towards Enhancing GQM. Proceedings of the Sixth European Software Cost Modelling Conference (ESCOM), May 1995.

[SM83] G. Salton, M. J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill Book Co., New York, 1983.

[SME92] Software Management Environment (SME) Concepts and Architecture -- Revision 1, NASA/GSFC Code 551, SEL Series, Report SEL-89-103, September, 1992.

[SPD+94] W. Schäfer, R. Prieto-Díaz, M. Matsumoto. Software Reusability. Ellis Horwood, 1994.

[SR92] C.B. Skalak, E. Rissland. Arguments and Cases: An Inevitable Twining. Artificial Inteligence and Law. An International Journal, 1(1), 1991.

[Ste00] STELA. Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, 2000 (www.eps.ufsc.br/stela.html)

[TA97a] C. Tautz, K.-D. Althoff. Using Case-based Reasoning for Reusing Software Knowledge. In Proceedings of the 2nd International Conference on Case-Based Reasoning, Providence, RI, July 1997.

[TA97b] C. Tautz, K.-D. Althoff. Operationalizing the Reuse of Software Knowledge Using Case-Based Reasoning. Technical Report IESE-Report No. 017.97/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern

(Germany), 1997.

[TG99] C. Tautz, C. Gresse von Wangenheim. A Representation Formalism for Supporting Reuse of Software Engineering Knowledge. Conference on Expertsystems XPS'99, Germany, 1999.

[TG98] C. Tautz, C. Gresse von Wangenheim. A Representation Formalism for Supporting Reuse of Software Engineering Knowledge. Technical Report IESE015.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, 1998.

[Tru97] D. Trump. Using the WWW and the Internet to Support Corporate Reuse. Lexis-Nexis Corporation. The Eighth Annual Workshop on Software Reuse, Ohio, 1997.

[TZ97] R. Tesoriero, M. V. Zelkowitz, The Web Measurement Environment (WebME): Combining and Modeling Distributed Data. In Proc. of the 22nd Annual Software Engineering Workshop, Goddard Space Flight Center, Greenbelt, Maryland, December,1997.

[Tve77] A. Tversky. Features of Similarity. Psychological Review, 84, 1977.

[Wat97] I. Watson. Applying Case-Based Reasoning Techniques for Enterprise Systems. Morgan Kaufmann Publisher, California, 1997.

[Wes95] S. Wess. Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik. Ph.D. Thesis, University of Kaiserslautern, Germany, infix Verlag, 1995.

[WS88] M. Wood, I. Sommerville. A Information Retrieval System for Software Components. ACM SIGIR Forum, 22(3/4), 1988. .

[ZS95] M. Zand, M. Samadzadeh. Software Reuse: Current Status and Trends. Journal of Systems and Software, 30(3), September 1995.

Apêndice A. Dados do Estudo Empírico

A Figura 25 mostra graficamente os dados plotados representando esforço total dos usuários gasto durante o planejamento do programa de mensuração (incluindo GQM2.2.2-GQM3.1).

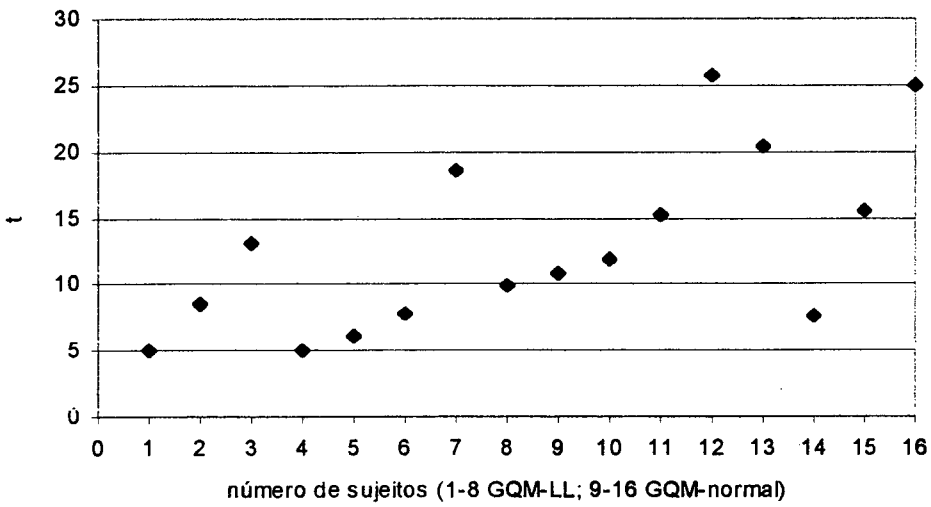


Figura 25 Dados do esforço total gasto

A Figura 26 mostra os dados para a percentagem de artefatos mal-definidos no plano do programa de mensuração (incluindo questões GQM, modelos de qualidade, medidas GQM e procedimentos de mensuração) em relação ao número total desses artefatos desenvolvidos por cada sujeito.

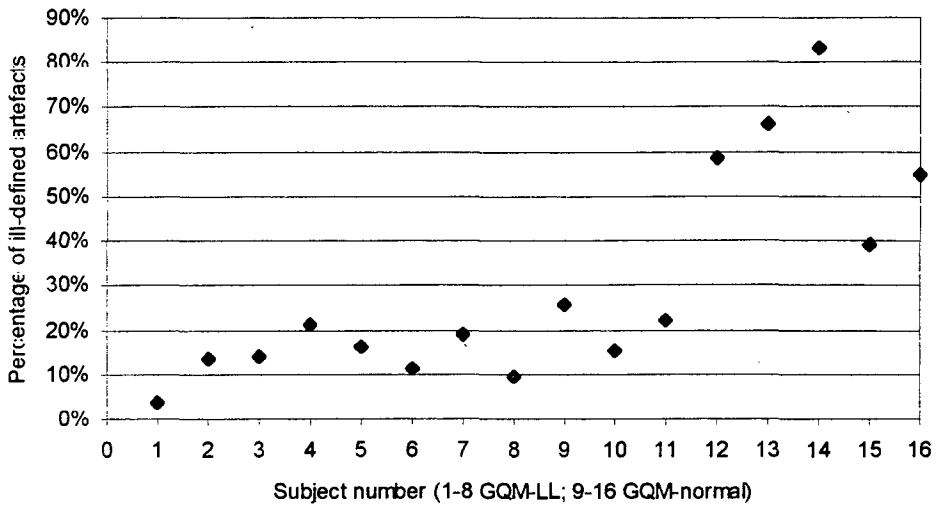


Figura 26 Dados para a percentagem de artefatos mal-definidos

Apêndice B. Publicações Relevantes

Case-Based Reasoning Approach to Reuse of Experiential Knowledge in Software Measurement Programs

C. Gresse von Wangenheim, A. R. Moraes, K. Dieter. Althoff, R. M. Barcia, R. Weber, A. Martins. Case-Based Reasoning Approach to Reuse of Experiential Knowledge in Software Measurement Programs. In Proceedings of the 6th German Workshop on Case-Based Reasoning, Berlin, Germany, March 1998.

Case-Based Reasoning Approach to Reuse of Experiential Knowledge in Software Measurement Programs

Christiane Gresse von Wangenheim[1], Alexandre Moraes Ramos[1], Klaus-Dieter Althoff[2], Ricardo M. Barcia[1], Rosina Weber[1], Alejandro Martins[1]

[1] Universidade Federal de Santa Catarina - Production Engineering
Florianópolis, Brazil
{gresse,amr,rbarcia,rosina,martins}@eps.ufsc.br

[2] Fraunhofer Institute for Experimental Software Engineering
Kaiserslautern, Germany
althoff@iese.fhg.de

For the successful application of innovative software engineering technologies in industry, the technologies have to evolve incrementally based on continuous feedback from practice. Experiences about their practical application have to be systematically collected and stored in corporate memories and reused in future software projects. This promotes the sharing of experiences across individuals and projects, the formulation of best practices and facilitates the successful application of tailored technologies in practice. This paper presents a case-based reasoning approach for capturing and reusing experiential knowledge on software measurement programs in industry. A representation structure for experiential measurement knowledge is described in detail and knowledge retrieval and acquisition techniques are presented.

1 Introduction

For the improvement of quality and productivity in software organizations, many software engineering technologies have been created during the last years. These technologies, in general, provide an explicit conceptual representation of the tasks to be performed. This representation is convenient for summarizing and communicating complex task knowledge. However, while transferring innovative software engineering technologies into industry, they have to be tailored to the specific characteristics and needs of a particular organization. Through continuous learning based on feedback from their application in practice, these technologies have to be developed in an incremental and evolutionary manner. Therefore, the technology representations are likely to be simplified, often do not include aspects of their application circumstances and knowledge on how to use these technologies in practice.

Recent studies [CM96] have proposed that "experiential knowledge" [Nor93], in form of past memories, is an additional important source of knowledge which contributes to learning. Experiential knowledge guides responding new situations based on similar past experiences. Thus, accompanying technologies by experiential knowledge, describing how the tasks have to be performed while taking into account specific goals and characteristics of a particular software project will substantially facilitate

their future application in practice. Reusing experiential knowledge can prevent the repetition of past failures and guide the solution of actually occurred problems. A decreased number of problems and their efficient solution will result in cost and time savings. Furthermore, the creation of organization-specific software competencies promotes the wide-spread effective use of innovative technologies in practice. Continuous feedback from their application helps the systematic enhancement and tailoring of software engineering technologies to better meet practical needs.

In order to operationalize this "learning from measurement experiences" in industrial environments, corporate memories for the systematic acquisition and the organization wide communication of this experiential knowledge have to be built [BM96,KS96, BCR94]. As a logical and physical structure for the continuous build-up of software know-how in an organization, the *Experience Factory* (EF) approach [BCR94] (see Figure 1) has been proven to be a successful solution. The experience factory approach introduces an infrastructure for analyzing and synthesizing all kinds of experiences, acting as a repository for those, and supplying these experiences to projects on demand. The main problems concerning the operationalization of the EF in practice are to capture experiences, to represent and store knowledge in a reusable form, and reuse efficiently and effectively this knowledge in future software projects. Newly gathered experiences have to be continuously acquired and integrated into the available knowledge. *Case-based reasoning* [AP94] appears to be the optimal approach [ABG98,GAB98,Hen97] for the operationalization of an experience base in practice. The major advantages of CBR in this context are the similarity-based retrieval for primarily experiential knowledge and its continuous incremental learning as an integrated part of the reuse process.

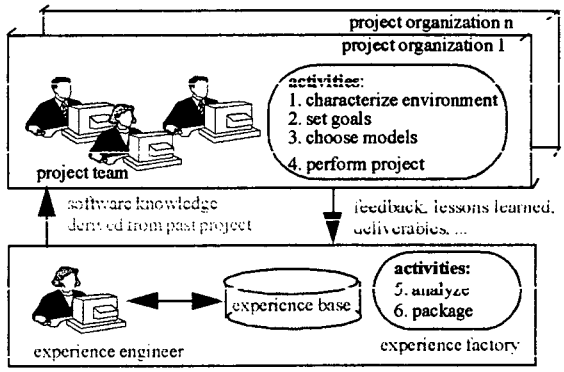


Figure 27: Experience Factory [BCR94]

We demonstrate our approach by using the Goal/Question/Metric Paradigm (GQM) [BDR97, BW84] as an innovative technology for software engineering measurement. GQM is a goal-oriented measurement approach, which helps defining and implementing operational and measurable software improvement goals. It has been successfully applied in several companies [CEM96,BCG92], such as NASA-SEL, BOSCH, Digital, and Schlumberger. Since it is an intellectually complex, resource-consuming task which requires experienced people, the availability of experiential measurement knowledge is expected to significantly contribute to the improvement of the creative process of planning measurement programs and lead to substantial effort reductions [GB97].

In this article we focus on experiential knowledge wrt. GQM-based measurement pro-

grams. A short introduction on GQM-based measurement and scenarios, illustrating the reuse potential, are given in Section 2. The case-based approach is presented in Section 3, addressing the representation of experiential measurement knowledge and techniques for the retrieval and acquisition of experiential measurement knowledge. Conclusions and future research directions are discussed in Section 4.

2 Application Domain: Software Engineering

The Goal/Question/Metric (GQM) approach is a technology for goal-oriented measurement in software projects. In GQM programs, the analysis task of measurement is specified precisely and explicitly by a detailed measurement goal, called GQM goal. Relevant measures are derived in a top-down fashion based on the goals via a set of questions and quality models. This refinement is precisely documented in a GQM plan, providing an explicit rationale for the selection of the underlying measures. The data collected is interpreted in a bottom-up fashion in the context of the GQM goal, questions and models, considering the limitations and assumptions underlying each measure. More information on GQM can be found in [BDR97,GB97,GHW95, BW84]. Here, the objective is to facilitate the establishment of measurement programs in practice by reusing experiential knowledge. The following scenarios illustrate how this knowledge can be used in order to support the GQM approach.

Scenario 1: Warning for potential failures

Tasks of the GQM process are complex and consequently error-prone to a high degree. Several critical problems can occur during its application in practice. Many problems could even be prevented, if their potential would be known in advance by the measurement responsible. Therefore, an overview on all experiences available on problems of a particular GQM task are provided to the measurement responsible before its execution. For example, during the development of the GQM plan, interviews are performed in order to acquire all relevant information wrt. the GQM goal. Before starting these interviews, the interviewer can request all experiential knowledge available on problems which occurred during this task in past measurement programs. For example: *The interviewee did not provide any information, because s/he did not know the objectives of the measurement program and which information was expected from her/him.* Knowing about problems occurred during this task in the past, will sensitize the responsible for potential problems, trying to prevent their repetition. Sometimes, problems which occur in subsequent phases of the measurement program, are due to failures during earlier phases. For example: *The development of the GQM plan was complicate, because only incomplete knowledge had been acquired during the interviews, e.g. the classification categories (low,medium,high) of the context factor "experience of developers" had not been defined. Consequently, additional follow-up interviews were necessary to precisely define the classification categories and their semantics, e.g., level of experience is high, if the developer works for more than 2 years in the development of telecommunication systems, before the development of the*

GQM plan could be continued. Providing a set of all experiential knowledge available on problems which were originated in the GQM task of interest, the measurement responsible will be aware of potential problems in advance. These experiences, providing knowledge beyond the scope of an individual or project will promote organizational learning concerning the application of software engineering technologies.

Scenario 2: Guiding the solution of a problem

When actually a problem occurs during the execution of a GQM task, its solution can be guided by experiential knowledge about similar problems and how they have been solved in past measurement programs. An example of a problem occurring during data collection is: *In the current measurement program invalid data on effort spent on software activities (e.g., hours spent on testing, hours spent on repair faults) has been collected by the project personnel.* Based on context characteristics of the problem situation (e.g. organization xyz) and the problem description (see above) relevant experiential knowledge is retrieved from the experience base and provided to the measurement responsible. Besides the context characteristics and the problem description, also the cause of the problem is important for the selection of an adequate solution strategy. Therefore, the causes of past problems, which have been explicitly captured in the past, are provided to the user. Based on these information s/he can explore the suggested reuse candidates in order to select the ones which fit best the actual needs and characteristics. For example, the cause of a similar problem in the experience base was: *Due to the weekly collection of effort data, it was difficult for the data collectors to reconstruct the time they had spent on particular activities each day.* If the actual problem was caused by a similar reason, knowledge on how the problem was solved in the past, e.g. *effort data was collected daily*, can guide the successful solution of the actual problem. An explicit description on the outcome of the solution applied on the past problem, can further indicate what worked and what did not in the particular environment, e.g. *then valid data was collected and the problem was successfully solved.* Storing also experiences regarding failed attempts to solve problems, provides additional information on potential failures aiming at their prevention in the future, e.g., *the incompleteness of the effort data collected increased, due to an increased data collection overhead through their daily collection. Furthermore the motivation and willingness of the data collectors decreased considerably.*

3 Case-Based Reasoning Approach to Capturing and Reusing Experiential Measurement Knowledge

For the experience-based support of GQM measurement programs, a case-based reasoning approach for the operationalization of experience bases for the capturing and reuse of measurement competencies in industrial environments is presented. Here, we focus on experiential knowledge gathered in individual past GQM-based measurement programs in practice. In the experience base (GQM-EKB)[1], experiential knowl-

edge on GQM measurement are stored as cases (GQM-PSE)[1], stating the problem occurred and the adopted solution strategy in the past. During the performance of new GQM measurement programs, GQM-PSEs are retrieved from the experience base, facilitating measurement. Based on context characteristics of the actual situation, e.g. name of organization, adequate cases with similar characteristics are retrieved from the experience base and provided to the user as reuse candidates. Through an interactive browsing and navigation system the user can explore the retrieved cases, select the most appropriate one, and if necessary, adapt the selected case appropriately to meet the specific needs of the actual situation. Since the GQM-EKB is used as a communication medium to share experiences organization wide, the retrieval and reuse of cases is emphasized, rather than their automated adaptation to specific characteristics of the actual situation. Integrated into the problem solving process is the acquisition of new experiential knowledge. Each time past experiences are reused in order to solve an actually occurred problem, new experiential knowledge is captured and available for problem solving.

In the following sections, we describe the representation of the experiential measurement knowledge, its retrieval and acquisition in detail.

3.1 Representation of Experiential Measurement Knowledge

As an important source for guiding the application of the GQM approach in practice, experiential measurement knowledge is captured in the GQM-EKB. Due to the specific nature of experiential knowledge, which supports the handling of exceptions, it is captured by stating an occurred problem during the performance of a GQM program and its solutions experienced in past software projects. Each case of the experience base is related to a specific problem. A GQM-PSE case includes a description of the problem, the adopted solution and information about the resulted outcome. In order to facilitate effective retrieval and provide detailed guidance for the acquisition of GQM experiences, these basic parts are refined into detailed dimensions and associated by a context description in order to allow the identification of relevant experiences in a particular environment. In addition, free-text descriptions (comments) are captured for each basic part in order to guarantee the comprehensive representation of the experiences beyond the defined dimensions. Basically, GQM-PSE consist of the following dimensions:

- **Context Description.** The organizational and project-specific context from which the experience originates is described (e.g. name of organization, programming language, application domain). In order to keep the context description minimal, only characteristics which are relevant to the particular GQM-PSE are listed. For exam-

1. This particular instantiation of an experience base is described by GQM-Experiential Knowledge Base (GQM-EKB).

1. Such a case describing a problem occurred during a GQM task and its solution is described by GQM-Problem Solution Experiences (GQM-PSE).

ple, the type of software (e.g. embedded software) might be irrelevant to a problem concerning the validity of data collection. Whereas, the duration of the software project or the size of project team might have an impact on the causes of the problem.

- **Problem**. The problem occurred during the GQM program is described.
- **Cause(s) of Problem.** The cause of the problem is explicitly described, if known. The objective is to prevent the repetition of potential problems in future measurement programs based on explicit knowledge on the causes originating the problems. As a problem can be caused through the interaction of several factors, for each cause detailed information is stated.
- **Solution.** The solution applied to solve the problem is described. Its guides coping with new problems while reusing past solution strategies in future measurement programs.
- **Outcome.** The resulted outcome of the solution applied is described in order to anticipate the expected outcome in future reuses of this case. Beside capturing successfully solved cases, also cases describing a failed tentative to solve a problem, are captured. These cases point out solutions which might potentially fail, when applied to solve the particular problem.
- **Basic information.** In order to support the appropriate usage of the available GQM-PSE and an easy and rapid selection of relevant ones, basic information on each GQM-PSE is provided:

The table below presents in detail the structure of GQM-PSE:

Case Structure	
Context Description	
Context characteristics	All relevant context characteristics are listed in form of attribute-value pairs[a]. For example, *((name of organization, xyz), (size of project team, 30 developers))*.
Comment	Additional information on the context description are stated as free-text.
Problem	
Problem description	The object affected by the problem and its state causing the problem are explicitly stated. The affected object can be any process, product or resource model or instance (e.g. data collection, effort data, tester). The state is described by listing the relevant attribute(s) of the affected object and its value[b]. For example, if invalid effort data is collected by project personnel, this can be described by *effort data: (validity, low)*.
Problem object type	The type of the objects affected by the problem is stated explicitly in order to facilitate retrieval. The objects are classified into processes, products or resources.
Problem task	The task in which the problem occurred is stated, e.g. *data collection*.
Problem roles	Roles of the software organization involved in the problem are listed, e.g., *developer*.
Goal unattained	The goal of the respective GQM task which has not been attained because of the problem is stated, e.g. *complete collection of valid data*.
Comment	Any additional information or comment on the problem is stated.
Cause(s) of Problem	
Cause description	The cause is described by the respective object and its state. For example, if the problem was caused due to the weekly collection, this is represented by *effort data collection procedure: (point in time, weekly)*.
Cause explanation	An explanation for each cause is provided in order to explain the relation between the problem and the stated cause. For example, *after one week it is difficult to remember the correct amount of effort spent.*
Cause task	The task causing the actual problem, which can be different from the task of problem occurrence, is identified, e.g. *development of measurement procedures*.
Cause role(s)	Roles of the organization involved in causing the problem are stated, e.g., *developer*.

Constraint(s)	Constraints wrt. the software project which influenced the problem, e.g., wrt. the availability of resources, effort, or duration, are stated. For example, *keep collection overhead less than 1% of total project effort.*
Comment	Any additional information or comment on the cause is added as free-text.
Solution	
Solution description	The solution is described by stating the modified, added or deleted object(s) and its state. For example, if the collection procedure has been modified to daily collection, this is described by *effort data collection procedure: (point in time, daily).*
Comment	Any additional information or comment on the solution is stated.
Justification	The solution is justified, focusing on the interdependencies between the cause, its explanation and the applied solution, e.g., *collection effort data in shorter time periods will reduce invalidity of data due to people forgetting details over time.* The justification allows the evaluation of the appropriateness of a past solution in the actual situation, while it provides an explicit rationale for its selection.
Outcome	
Outcome description	The results of the solution applied are described. It describes the state of the object stated in the problem description after the application of the solution. If, in addition, the state of other relevant objects changed due to the solution, e.g. caused new problems, these objects and their states are added. For example, if after the modification of the collection procedure, valid data is collected, this is stated by *effort data: (validity, high).*
Outcome assessment	The assessment states explicitly if the problem was successfully solved by the solution or failed, e.g. *solution was successful.*
Comment	Any additional information or comment on the outcome is described.
Failure explanation	If the applied solution failed to solve the problem, an explanation is given on why the goal of the respective task was still not achieved. For example, *because of a considerable increased effort due to the daily collection, the data collectors did not submit all required data.*
Next PSE	If the applied solution failed to solve the problem, the next attempt to solve the problem, stored as a new case in the experience base, is referenced, e.g. *case_43.*
Basic information	
Viewpoint	The role from which the knowledge was acquired is stated, e.g., *measurement expert.*
Representativeness	The representative of the GQM-PSE is given in terms of the number of individual software projects from which it was derived. For example, once captured from one software project a GQM-PSE can be confirmed in other projects, when it is reused which increases its representativeness.
Adm. information	such as creating date, ownership, access rights.

a. Attribute-value pairs are notated by (attribute, value).
b. An object and its state are notated by object: (attribute, value).

Beside the representation of explicit examples of problems and their solutions from individual projects, the representation of general domain knowledge, e.g., taxonomies and glossaries, can further facilitate the acquisition and reuse of experiential knowledge. Taxonomies represent ordered arrangements of entities according to their presumed relationships, e.g., a hierarchical taxonomy of organizational units (see Figure 28). Glossaries define organization-specific terminology and basic concepts. This domain knowledge can guide and direct the appropriate specification of relevant objects, e.g. roles, tasks. Glossaries support the adequate use of terms, their consistency across an organization, and prevent misunderstandings and communication problems between different roles, e.g., developers and senior managers.

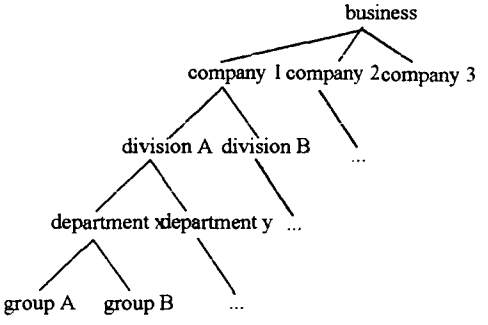


Figure 28: Organization taxonomy

3.2 Retrieval of Experiential Measurement Knowledge

The establishment of GQM measurement programs in practice is facilitated by reusing experiential knowledge. Each task of a GQM measurement program is supported by providing experiences from past measurement programs on request in order to anticipate potential problems or to solve existing ones. Based on initially given characteristics of the actual situation, similar cases are retrieved from the experience base and suggested to the user as reuse candidates. The user can explore these reuse candidates through browsing and navigation and concentrate on the ones which fit best the current needs. Depending on the objective, to prevent failures or to solve an existing problem, two applications are identified.

Application1: Warning for potential failures

Before starting a GQM task, the user can request an overview on failures occurred in past measurement programs, which were originated in this particular task (see section 2/scenario 1). The system guides the elicitation of relevant context characteristics of the actual project, e.g. name of the department, and the GQM task of interest, e.g., data collection, by questioning the user. Relevant reuse candidates are searched by exact matching of the GQM task of current interest and the Cause task of the case available in the experience base. Reuse candidates with similar context characteristics are retrieved under consideration of the characteristics describing the actual context given by the user. For example, given a specific department as organizational scope of interest, experiences gathered in this particular department are more similar to the current situation, than experiences gathered in other departments or even across companies. Since, the objective of this application is to provide an overview on all potential failures originated in the GQM task of current interest, the following information is extracted of the retrieved cases and provided to the user:

- context descriptions and basic information to allow the user to examine the validity of the proposed case in the actual situation,
- detailed descriptions of causes of problems, pointing out possible failures during the task of interest, which may cause a problem during this or a subsequent task,
- detailed descriptions of the problems occurred, anticipating what could be provoked by the failure.

Further information on the retrieved GQM-PSEs, or other GQM-PSEs available in the experience base can be explored by the user via browsing and navigation along references between individual cases.

Application2: Guidance of solution of problems

When a specific problems occurs during the GQM process, the user can request help to guide its solution by reusing solutions of past, similar problems (see section 2/scenario 2). Based on a description of the actual problem, the task when the problem occurred and specified context characteristics, a set of similar problems is retrieved from the experience base. A set of adequate reuse candidates is provided to the user, show-

ing the following information:

- context description, basic information, problem and its cause(s) in order to allow the user to evaluate in detail the validity of the suggested case in the current situation,
- solution in order to guide the solution of the actual problem, by transferring and, if necessary adapting the past solution to the actual situation. The justification for the application of the solution allows the user to evaluate the appropriateness of the suggested solution in the actual situation.
- outcome in order to inform the user about the expected consequences of the application of the solution, e.g. if it is expected to solve the problem successfully or might cause other problems.

While proposing the case to the user, s/he can further explore the dimensions of the retrieved cases, e.g, additional comments, and related cases in order to make informed decision concerning the solution of the actual problem. If necessary, suggested solutions are adapted by the user to meet the current needs.

3.3 Acquisition of Experiential Measurement Knowledge

Essential for continuous improvement of software engineering technologies is their incremental evolution based on feedback from industrial applications. Consequently, the knowledge in the experience base has to be enhanced and updated each time a new measurement program is established. Each time a problem occurs during a GQM measurement programs, it is captured as a new GQM-PSE. The acquisition of new experiences is intertwined into the problem solving process. While the user requests experience-based support for the solution of an occurred problem, the initially given problem description and context characteristics used for retrieval of similar cases are in parallel captured documenting a new GQM-PSE. If cases retrieved from the GQM-EKB are used for the solution of the actual situation, the reused case serves as a basis for the further documentation of the actual situation. The description of the new case is carefully reviewed by the user and missing information is added and deviations from the reused case are modified. For example, if the same problem occurs wrt. the collection of fault data instead of effort data, the case description has to be modified wrt. the type of data collected. The acquisition of new experiences is guided through the detailed case structure, which explicitly addresses relevant dimensions. The usage of organizational domain knowledge, such as glossaries and taxonomies, further facilitates the consistent description of experiences across individuals and projects.

4 Conclusions

For the successful application and continuous evolution of software engineering technologies in practice, experiential knowledge has to be captured in corporate memories and reused in future applications. In this paper, we present a case-based reasoning approach for the operationalization of experience bases on experiential measurement knowledge. We describe the representation structure of the experiential knowledge

and techniques for the retrieval of adequate reuse candidates and the acquisition of new experiences. Currently, we are implementing a an experience base for the experience-based support of the planning of GQM measurement programs by using a CBR-tool [Tec97]. The application of the approach offers the possibility of systematic acquisition of experiential measurement knowledge in practice and, therefore, provides a basis for further research on the evolution and generalization of technologies wrt. packaging and reuse of experiential knowledge on software engineering technologies, based on feedback from its use in practice.

5 References

[ABG98] K.D. Althoff, A. Birk, C. Gresse von Wangenheim, C. Tautz. CBR for Experimental Software Engineering. In H. D. Burkhard, M. Lenz et al., eds., Case-Based Reasoning Technology from Foundations to Applications, Springer, Berlin, to appear 1998.

[AP94] A. Aamodt, E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, vol. 17, nr. 1, 1994.

[BM96] J.M. Barr, R.V. Magaldi. Corporate Knowledge Management for the Millennium. In I. Smith and B. Faltings, ed., Advances in Case-Based Reasoning, Springer, Berlin, 1996.

[BCG92] V. R. Basili et al. The Software Engineering Laboratory-An Operational Software Experience Factory. In Proceedings of the 14th Int. Conference on Software Engineering, 1992.

[BCR94] V. R. Basili, G. Caldiera, H. D. Rombach. Experience Factory. In John J. Marciniak, editor, Encyclopedia of Software Engineering, vol. 1, pp. 528-532. John Wiley & Sons, 1994.

[BDR97] L. C. Briand, C. M. Differding, H. D. Rombach. Practical Guidelines for Measurement-Based Process Improvement. Software Process Improvement and Practice, to appear 1997.

[BW84] V. R. Basili, D. M. Weiss. A Methodology for Collecting Valid Software Engineering Data. IEEE Transactions on Software Engineering, SE-10(6):728-738, 1984.

[CEM96] CEMP Consortium. Customized Establishment of Measurement Programs. Final Report, ESSI Project Nr. 10358, Germany, 1996.

[CM96] JR. Cho, R.C. Mathews. Interactions Between Mental Models Used in Categorization and Experiential Knowledge of Specific Cases. The Journal of Experimental Psychology, 49A (3), 1996.

[GB97] C. Gresse, L.C. Briand. Requirements for the Knowledge-Based Support of Software Engineering Measurement Plans. In Proceedings of the 9th Int. Conference on Software Engineering and Knowledge Engineering, Spain, 1997.

[GAB98] C. Gresse von Wangenheim, K-D. Althoff, R. M. Barcia, C. Tautz. Evaluation of Technologies for Packaging and Reuse of Software Engineering Experiences. Submitted to Int. Conference on Industrial Engineering Applications of Artificial Intelligence and Expert Systems, 1998.

[GHW95] C. Gresse, B. Hoisl, J. Wüst. A Process Model for GQM- Based Measurement. Technical Report STTI-95-04-E, Software Technology Transfer Initiative, Germany, 1995.

[Hen97] S. Henninger. Capturing and Formalizing Best Practices in a Software Development Organization. In Proceedings of the 9th Int. Conference on Software Engineering and Knowledge Engineering, Spain, 1997.

[KS96] H. Kitano, H. Shimazu. The Experience-Sharing Architecture. In D. Leake, ed., Case-Based Reasoning Experiences: Lessons Learned & Future Directions, 1996.

[Nor93] D.A. Norman. Things That Make Us Smart: Defending Human Attributes in the Age of the Machine. Reading Ma, Addison-Wesley, 1993.

[Tec97] CBR-Works. TecInno GmbH, Germany (http://www.tecinno.de/tecinno_e/ecbrwork.htm)

Case-Based Reuse of Software Engineering Measurement Plans

C. Gresse von Wangenheim, A. von Wangenheim, R. M. Barcia. Case-Based Reuse of Software Engineering Measurement Plans. In Proceedings of the 10th International Conference on Software Engineering and Knowledge Engineering, California, June 1998.

Proceedings of the 10th International Conference on Software Engineering and Knowledge Engineering, California, June 1998

Case-Based Reuse of Software Engineering Measurement Plans

Christiane Gresse von Wangenheim,
Aldo von Wangenheim, Ricardo M. Barcia

Universidade Federal de Santa Catarina
Florianópolis, Brazil
{gresse,rbarcia}@eps.ufsc.br,awangenh@inf.ufsc.br

One essential infrastructure for software process improvement are measurement programs. Their planning and successful implementation requires, in practice, a significant amount of effort and expertise. Cost may be reduced and quality of measurement may be improved by providing knowledge-based support and reusing experiences gathered from past measurement programs. For the systematic improvement and organization-wide communication of measurement know-how, organization-specific experiences to be collected systematically, stored in corporate memories and reused in future software projects. Therefore, learning knowledge-based systems have to be operationalized in industrial environments. This paper presents a case-based reasoning approach for capturing and reusing experiences on software measurement programs. A representation structure for measurement experiences is defined and knowledge retrieval, acquisition and integration techniques are described.

Keywords: organizational learning, software process improvement, case-based reasoning, software measurement, Goal/Question/Metric Paradigm.

6 Introduction

For the continuous improvement of software quality and productivity within organizations, capturing and reusing explicit software engineering technology know-how tailored to the specific characteristics and needs of a particular organization is essential [1]. A fundamental infrastructure for the derivation of relevant quantitative and qualitative data on software processes and products is the establishment of goal-oriented measurement programs in software projects focusing on the specific business goals and characteristics of the company [1]. In this context, the Goal/Question/Metric Paradigm (GQM) [2, 3] is an optimal approach that helps defining and implementing operational and measurable software improvement goals. The approach supports the top-down definition of measures based on measurement goals and the bottom-up interpretation of the data collected in the context of each measurement goal. It has been successfully applied in various companies, such as NASA-SEL, Robert Bosch, Digital, and Schlumberger. To be effective and efficient, measurement programs have to be tailored to the characteristics of the specific organization, their software processes, products and business goals. Therefore, one of the major success factors of a measurement program is its appropriate planning. Since it is an intellectually complex process which requires experienced people, it is usually costly and the likelihood of committing mistakes is high. Hence, the availability of explicit measurement knowledge tailored to the organization specific characteristics and needs, can significantly contribute to the improvement of planning of measurement programs and lead to a substantial effort reduction [4]. Therefore, experiences on applying measurement programs have to be gathered under consideration of characteristics and business goals of the particular organization to enable their adequate reuse in future measurement programs.

This framework of experiences from past software projects[1], denoted as *experienceware*, is an important source of knowledge which contributes to learning [5]. Experienceware guides responding new situations based on similar past experiences in the context of an organization, including expertise and lessons learned on software engineering technologies, quality models, and deliverables. In the context of GQM-based measurement programs this refers to all related products, from measurement goals to data collection instruments. Although there is often a need to tailor these products to specific project characteristics, they are reusable to a large extent [4]. Reusing products which have been developed in the same organizational context will require less effort and will be more likely to address the specific needs. Reusing lessons learned helps to improve the planning of measurement programs over time by preventing the repetition of failures and guiding the effective and efficient solution of problems. The availability of experienceware on measurement generally facilitates the organization-wide application of measurement programs. Furthermore, it supports the continuous evolution of the measurement technology through feedback explicitly gathered to better fit practical needs. However, the complexity of measurement plans makes the understanding and the identification of relevant and reusable measurement products already developed in the company's context difficult. This is exacerbated by the complex net of interdependencies between GQM products. Sophisticated ways are required of storing knowledge to allow intelligent search, acquisition of new experienc-

1. In the context of this paper, experiences specific to measurement planning.

es, and navigation in knowledge bases. Thus, corporate memories for the systematic acquisition and organization wide communication of these experiences have to be build, operationalizing «learning from experiences» in industrial environments [1, 6].

As a logical and physical structure of corporate memories for the continuous build-up of software know-how in an organization, the *Experience Factory* (EF) approach [1] has been proven to be a successful solution. The experience factory approach proposes an infrastructure for analyzing and synthesizing all kinds of experiences, acting as a repository for those, and supplying these experiences to projects on demand. To operationalize this framework, technologies for capturing experienceware on software engineering technologies, the representation and storing of experienceware in a reusable form and the efficient and effective reuse of this knowledge in future software projects have do be developed. New gathered experiences have to be continuously acquired and integrated into the available knowledge base. To achieve this goal, a learning knowledge-based system has to be build as an integrated support platform, operationalizing the experience factory approach in industry. *Case-based reasoning* (CBR) [7] is an optimal approach for the development of an experience base in practice and provides a broad support for the development of knowledge-based systems [6]. The major advantages of CBR in this context are similarity-based retrieval for all kinds of artifacts and its primary focus on experiential knowledge. Continuous incremental learning occurs as a natural by-product of problem solving by revising and capturing experiences each time a new problem is solved.

7 Reuse of GQM Measurement Plans

The Goal/Question/Metric approach (GQM) [2,3,4,8] is a specific technology for goal-oriented measurement in software projects, supporting the definition and implementation of operationalizable software improvement goals. Based on a precisely and explicitly specified measurement goal, relevant measures are derived in a top-down fashion via a set of questions and quality/resource models. This refinement procedure is precisely documented in a GQM plan, providing a rationale for the selection of the underlying measures. Data is collected wrt. the measures and interpreted in a bottom-up fashion in the context of the models, question and measurement goals, while considering the limitations and assumptions underlying each measure. The establishment of measurement programs in practice has to be facilitated by reusing measurement experienceware. The following scenario illustrates how experienceware gathered from past projects can be used in order to support the planning of GQM-based measurement programs. Suppose a scenario where a company, IntelliCar, which produces embedded software for automobiles has two main software development departments: department FI which develops software for fuel injection devices and department ABS which develops software for ABS brake control devices. As this company produces embedded software, one of its most important goals is to produce zero-defect software. Therefore, department FI established successfully a quality improvement program based on measurement two years ago. Several measurement programs have been performed in different software projects focusing on the characterization and improvement of the reliability of the software process. Now, also department ABS wants to start measurement-based improvement. As the contexts of the projects in both departments are similar and the improvement goal is the same, experiences available in department FI can be reused in order to reduce the planning effort and to improve the quality of the measurement program in department ABS.

First, a goal to be achieved by the measurement program is explicitly defined by describing the *object, purpose, quality focus, viewpoint* and the *context* of the measurement program. Based on the improvement goal «*improve the reliability of the software system*» and the specific context characteristics, the measurement goal of the project in department ABS is defined as «*Analyze the software development process in order to characterize and improve the reliability from the viewpoint of the software developer at ABS/IntelliCar*». Concerning this measurement goal, relevant quality aspects and influence factors on these aspects have been acquired during interviews with the developers of department ABS. These are represented as a set of questions operationalizing the goal in the GQM plan, as shown in Figure 1. In order to operationalize the ques-

Q7. What is the total number of defects detected before delivery?
Q8. What is the distribution of faults by life cycle phase of detection before delivery?
Q9. What is the distribution of defects?
Q10. Does the type of inspections have an impact on the effectiveness of inspections?
Q11. Does the experience of the developers have an impact on the number of faults introduced into the system?

Figure 1. Examples of questions of the GQM plan

tions of the GQM plans, to quantify the various abstract attributes of the object being studied, and to define precisely how quality comparisons, evaluations, and predictions are to be performed [2], quality models for department ABS have to be developed. These models define explicitly the quality attributes wrt. the questions of the GQM plan in the particular context. Assume, for example, that the question «*Q4. Does the*

Descriptive Model: Effectiveness of inspections	
Context:	company IntelliCar, department FI
Assumptions:	The defect density is comparable across documents.
Model description:	effectiveness=(number of defects detected during inspections)/size of document
Attributes:	number of defects detected during inspections; size of documents
Measures:	count of critical defects logged during the meeting; number of operations specified

Figure 2. Example of quality model

type of inspection have an impact on the effectiveness of inspections?», was already evaluated in a similar measurement program in department FI. The applicability of the related quality model, as shown in Figure 2, in ABS's project can be directly assessed based on its underlying assumptions. If necessary, the model has to be adapted to characteristics of the specific project. For example, assuming that inspectors capabilities vary extensively between departments, the effectiveness of inspections is expected to depend not only on the size of the inspected document (as stated in the reused model), but

also on the training of inspectors. Then this new factor has to be included in the model.

Consider ABS's question «*Q1. What is the total number of defects detected before delivery?*». Specific terms are used in its formulation, e.g. the term *defect*. As quality models have to take into account the particular environment, the specific standards and terminology have to be well known. This can be supported through the (re-) use of organizational glossaries (see Figure 3).

Error: Error refers to human action that results in software containing a fault.
Fault: Fault is a condition that causes the software to fail to perform its required function.
Failure: Failure is the inability of a system or a component to perform a required function according to its specifications
Defect: used as synonym for fault

Figure 3. Excerpt of glossary

But there are also differences between the departments FI and ABS and not everything can be transferred straightforwardly. During the development of a quality model for question «*Q5. Does the experience of the developers have an impact on the number of faults introduced into the system?*», it comes out that no similar question has been raised before. The development of a quality model from scratch for this question can be facilitated by (re-)using a template for quality models based on the ones already used in IntelliCar. The template describes the structure of a quality model. For example, for the comprehensive definition of a quality model, the related context, the underlying assumptions, the model, the quality attributes and the respective measures have to be defined.

While defining a quality model for question «*Q3. What is the distribution of defects?*», it turned out that an operational refinement of the question is not possible due to missing information concerning the kind of distribution. The solution of this problem can be guided by GQM-Problem Solution Experiences (GQM-PSE) [4,9]. GQM-PSEs provide knowledge about problems which occurred during past measurement programs, how they have been solved and the extent to which those solutions strategies have been successful. For example,

Context	*organization IntelliCar; department FI*
Problem	*Question of the GQM plan cannot be refined into an operational quality model due to missing information.*
Cause of Problem	*During the interviews the necessary knowledge has not been acquired completely from the project personnel.*
Solution	*A follow-up interview was performed with the person(s) who mentioned the respective quality aspects during the first interviews in order to clarify the formulation of the GQM question.*
Outcome	*The required knowledge was acquired completely and the respective quality model was defined.*

Figure 4. Simplified example of GQM-PSE

if a similar problem already occurred during a measurement program in the department FI (see Figure 4), its respective solution can indicate the repetition of interviews in order to acquire completely the necessary information.

In addition, the refinement of the abstract concept «*distribution of defects*» as stated in question Q5 can be driven by using a taxonomy representing generalization relations between quality concepts in the specific environment (see Figure 5). For example, the taxonomy can guide the derivation of the appropriate quality model by selecting the kind of defect (e.g. fault, failure) department ABS wants to measure, and the type of distribution (e.g. per phase of detection, per detection mechanism). This approach of reusing experiences concerning other phases of the planning process can be supported correspondingly by reusing GQM experienceware.

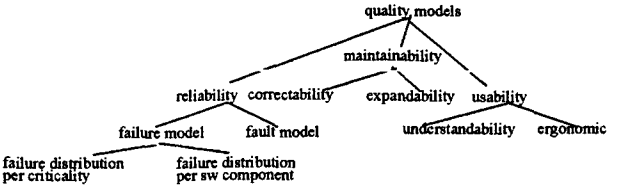


Figure 5. Taxonomy of quality models - example

8 Experience-Based Measurement Planning

In this section we present a case-based reasoning approach on the operationalization of a learning experience base for the systematic acquisition and organization-wide communication of measurement experienceware in industrial environments to facilitate the planning of GQM-based measurement programs. Regarding the specific requirements to an integrated support platform for the experience-based support of GQM planning, case-based reasoning (CBR) [7] provides an optimal approach for the operationalization of the reuse of organization-specific experienceware by providing techniques for problem solving and incremental sustained learning. The major advantages of CBR in this context are similarity-based retrieval for all kinds of artifacts and its continuous incremental learning as integrated part of the reuse process. Beside its primary focus on experiential knowledge, a particular advantage of CBR is that it can be extended using additional knowledge representation and learning capabilities (e.g., inductive learning of decision trees [10]).

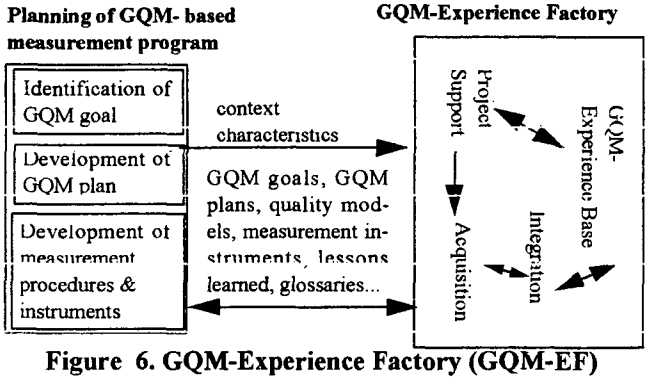


Figure 6. GQM-Experience Factory (GQM-EF)

Measurement experienceware is accumulated in the *GQM-experience base* (GQM-EB)[1], reflecting organization specific measurement know-how which has been gathered during past measurement programs. These experiences can be inquired during the planning of a new measurement program to find relevant know-how to guide, support and improve the present planning process. Relevant experiences are identified based on context characteristics, such as the similarity between the

1. Here, we consider a specific instantiation of the experience base focusing on experienceware on the planning of GQM-based measurement programs.

contexts of departments IF and ABS above, problems and goals of the software projects. Relevant reuse candidates are suggested to the user via a navigation system, which allows the interactive exploration of the candidates. From the set of retrieved reuse candidates, the most appropriate one is selected by the user, used as an initial basis for the actual planning process and, if necessary, adapted adequately to meet the specific needs of the current software project. Since the GQM-EB is used as a communication medium to share experiences organization-wide instead of providing a ready solution, the retrieval and reuse of experiences is emphasized, rather than their automated adaptation to specific characteristics of the present situation. New experiences are acquired and integrated into the GQM-EB, each time a measurement program is planned as an inherent part of the retrieval and reuse process.

8.1 Representation of GQM Experienceware

For the experience-based support of the planning of GQM programs in practice, measurement experienceware is represented in the GQM-EB [4]. In order to provide effective support, the GQM-EB has to capture various types of measurement experienceware [4] (see Figure 8). This includes knowledge on specific products developed during the planning of measurement programs, lessons learned wrt. the planning process, as well as, terminology and measurement concepts used in the specific organization and their interdependencies. In order to

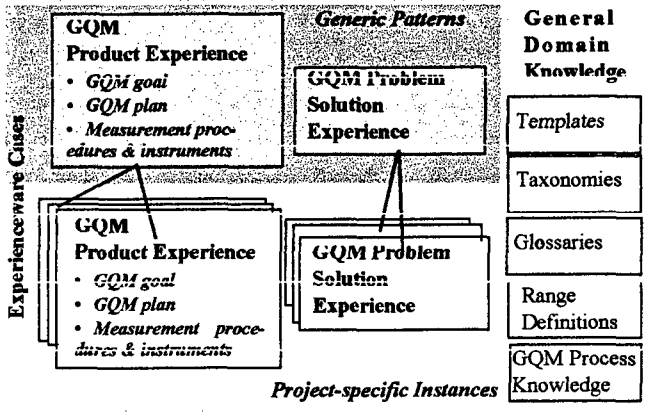


Figure 8. Types of experienceware

be effective, GQM experienceware has to be gathered in the specific organizational environment. In the context of software measurement, each software project contributes to the organizational software know-how by capturing specific experiences gathered in this project. The project-specific experiences are accompanied with a characterization of the particular project context which allow the retrieval of experienceware from similar software projects. Therefore, knowledge is primarily represented in form of concrete cases [7] which are context-specific descriptions of particular experiences gained during the planning of past measurement programs. To broaden the scope of application, generic patterns are represented, subsuming common project-specific experiences. These experiences are further enhanced by general domain knowledge on measurement terminology and concepts for additional support.

8.1.1 Experienceware Cases

The main form of knowledge representation are *cases* [7] representing project-specific measurement experiences. Two major types of experienceware cases are needed:

- GQM Product Experience (GQM-PE): These cases include all GQM products developed during the planning of a GQM-based measurement program. They are reused in similar software projects as a basis for the development of respective products. This results in a reduction of planning effort and improvement of the quality of the GQM products wrt. their reliability, completeness and consistency.
- GQM-Problem Solution Experience (GQM-PSE): They explicitly capture problem solution strategies that have been adopted on past measurement programs, their context of use, and information regarding their degree of success (see Figure 4). This can warn for potential failures in advance and support the finding of a solution fitting the application context and help set reasonable expectations.

8.1.1.1 Structure of GQM Product Experience

A GQM-PE represents all GQM products developed in a particular measurement program. It includes [4]:

- GQM measurement goal, describing the goal to be achieved by the measurement program.
- Abstraction sheet, documenting the knowledge acquired during the interviews as a basis for the GQM plan.
- GQM plan, including questions, quality and resource models and measures,
- Measurement procedures and instruments, defining the measurement procedures for the measures defined in the GQM plan and the instruments by which they are collected.

The GQM products are explicitly documented in the GQM-Product Experience Case. The knowledge concerning the GQM products is structured by a set of relevant attributes, e.g. the GQM goal is described through *object, purpose, quality focus, viewpoint* and *context*. Interdependencies between GQM products and/or their attributes are explicitly modeled, e.g. between measures and measurement instruments. Modeling various types of relationships allows the representation of the complex interdependencies of GQM products.

For the retrieval of GQM-PEs of similar software projects, an appropriate and unambiguous characterization of the environment from which the case has been obtained is essential. Characterizing the environment is important to relate the GQM products to the appropriate context from which they originate. The context of the experiences has to be described by a minimal set of characteristics, which allows to identify similar projects and to discriminate between them in order to find adequate reuse candidates. Examples of context characteristics are [4] business sector, improvement goals of the organization, and development process used. In order to facilitate the appropriate usage of available experiences, the GQM products of the case are further associated with basic information [4]. This information helps to assess the reuse potential of the case or parts beforehand, preventing the reuse of inadequate experiences. For example, if the experience was acquired in a

different kind of software project, differences in the terminology used could cause misunderstandings. Examples of basic information are: *viewpoint, acquisition technique, representativeness* and *administrative information*. In order to continuously improve the reuse of experiences and facilitate their adaptation to specific project characteristics, information about their reuse in measurement planning are captured explicitly [4], such as *preconditions for reuse, required adaptations, cost of reuse, dates of reuse* and *guidelines of reuse*.

8.1.1.2 Structure of GQM Problem-Solution Experience

To explicitly capture problem solution strategies on problems that occurred during the planning of past measurement programs, GQM-PSEs are modeled and represented as cases [9] concerning problems encountered while developing or using the GQM products (see Figure 4). Due to the specific nature of experiential knowledge, which supports the handling of exceptions, it is captured by stating an problem occurred during the planning of a GQM program in the past and the solution(s) adopted. Each case of GQM-PSE is related to a specific problem experienced in the past. Basically, a GQM-PSE case includes a context description, the problem occurred during the planning of a GQM program, a list of causes of the problem, the solution applied, the resulted outcome of the solution and basic information.

8.1.2 Generic Patterns of Experienceware Cases

Besides capturing project-specific instances of experienceware cases (GQM-PE and GQM-PSE) in the experience base, instances which share similar properties are organized under a more general structure. Experiences of measurement programs with similar characteristics and goals are clustered in generic patterns of measurement experiences. The generic patterns are structured as the project-specific cases, but contain knowledge on a higher level on abstraction. Abstraction is achieved by reducing the degree of detailness of the experiences by removing certain features or relations, which are not shared by all subsumed project-specific instances [11]. Through the synergetic combination of knowledge on different levels of abstractions in the GQM-EB the effectiveness of reuse can be increased by broadening the application spectrum. This also promotes the formation of generic patterns emphasizing important aspects across various measurement programs in similar contexts.

8.1.3 General Domain Knowledge

In order to support the appropriate reuse of experienceware and facilitate the consistent acquisition of new experiences across software projects, general domain knowledge wrt. the GQM experienceware in a particular environment is represented in the GQM-EB [4]. This includes the description of relevant parts of the organizational domain:

Templates of GQM products, e.g., the GQM goal template (see Table 15), represent the structure and expected content of GQM products. The (re-)use of templates can significantly reduce effort and risks and help to achieve a consistent view across various projects.

Taxonomies represent ordered arrangements of entities according to their presumed relationships, e.g., see Figure 5.

Dimension	Definition
Object	What will be analyzed?
Purpose	Why will the object be analyzed?
Quality Focus	What property of the object will be analyzed?
Viewpoint	Who will use the data collected?
Context	In which environment does the analysis take place?

Table 15: GQM goal template summary [2]

Taxonomies of entities related to measurement can be used for guiding the appropriate refinement of objects of interest.

Glossaries define a terminology and basic concepts related to GQM products in a specific environment (see Figure 3). A glossary supports the adequate use of terms, their consistency across an organization, and ensures that the reuse of GQM products is based on sound assumptions.

Range Definitions define precisely the ranges of attributes of cases in a specific organization. For example, measurement maturity may be classified at IntelliCar into: *high*-measurement established for 2 years, *medium*- measurement established at least once, *low*-never done measurement. The explicit definition of ranges of attributes supports the formulation of requests for reuse and the consistent description of new experiences across projects.

8.2 Experience-Based Support of GQM Planning

The planning of GQM measurement programs is supported by reusing experienceware of past measurement programs. While the GQM responsible is planning a GQM-based measurement program, s/he can request experience-based support. During each task of the planning process, appropriate cases describing past experiences from similar software projects are retrieved from the GQM-EB. The ones which best suits the current needs are used as an initial basis for the development of the GQM products or to support the performance of the task in the present measurement program.

Due to the fact, that each software project is different, no experiences concerning identical projects can be expected in the GQM-EB. Therefore, experienceware of «similar» projects has to be identified to support the current planning process. For example, while developing a quality model on the effectiveness of inspections, similar models from past measurement programs can serve as a starting point and be further tailored to the specific needs of the current project. In order to identify appropriate GQM experienceware in the GQM-EB, the present environment has to be characterized in terms of relevant characteristics of software project, organization and measurement program, e.g. see Figure 7. The elicitation of the context characteristics is interactively guided by the GQM-EF which provides characterization templates to the user. The templates are based on characteristics used in the past to describe the environment and evolve continuously as the user can add or remove characteristics when they become relevant/irrelevant for the characterization. In order to reduce the effort related to the elicitation of characteristics, the system infers supplemental values based on the ones stated. As for example, characterizations of a particular organization are available in cases in the GQM-EB (e.g. describing its staff size, application domain), these characterizations can be derived based on the

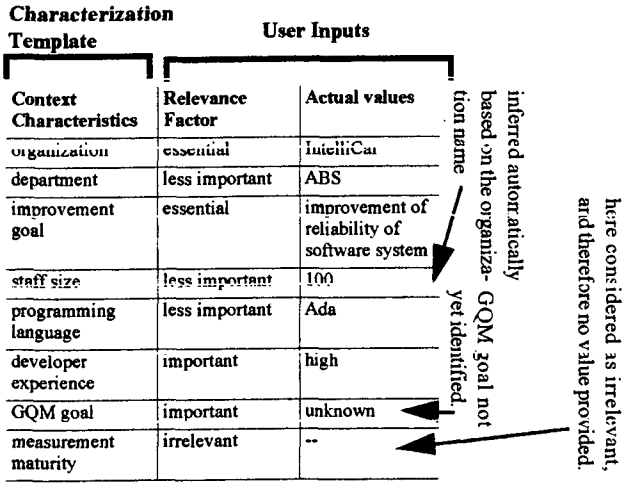


Figure 7. Simple example of characterization

name of the organization. The input of context characteristics can further be supported by general domain knowledge. Glossaries can be used for a consistent usage of terminology across projects and taxonomies guide and direct the appropriate definition of characteristics. Range definitions ease stating the present values and guarantee a consistent description across projects.

For the description of the present situation, certain characteristics of the context description might be absolute essential, others might be of less importance. In order to reflect the relevance of characteristics for the description of the current context, a *relevance factor* is elicited for each characteristic (see Figure 7). The relevance factor declares the importance of the particular attribute for the determination of the similarity between the actual context and the context description of a case in the GQM-EB. Here, e.g., the relevance factors are classified into *essential, important, less important,* and *irrelevant*. Accordingly to the relevance factor, each context characteristic can be associated with a weight used for the computation of the similarity through adequate CBR techniques [10].

Retrieving relevant experienceware cases

When experienceware of past measurement programs are reused in order to facilitate the planning of the actual one, appropriate cases have to be identified in the GQM-EB. As the usefulness of past experiences can only be determined when tried to be reused in the current project, the a posterior criteria of usefulness is reduced to the a priori criteria of similarity [10] between the context characterization of the cases, assuming that the similarity of context characterizations also implies the usefulness of the solution. This is a non trivial task, as it considers the evaluation and comparison of complex knowledge representations. Furthermore, it includes the handling of incomplete information, as in the software measurement domain certain context characteristics might be still unknown, when initiating a measurement program, e.g. size of system or GQM goal (see Figure 7), or cases in the GQM-EB might be described incompletely, because certain context characteristics were not reported in the past.

For the retrieval of reuse candidates, the GQM-EB is first searched by exactly matching the current context characteristics marked as essential and the respective ones of the cases stored in the GQM-EB. The resulting set of cases is further evaluated wrt. their degree of similarity regarding the specified characteristics in the input marked as important or less important. Characteristics, which are not specified or marked as unimportant are disregarded for the calculation of similarity. Figure 8 shows a simplified example: while comparing the cases of the GQM-EB with the given context characterization of the present situation, case PE_003 and case PE_007 are retrieved as reuse candidates, because the values of the attributes marked as essential are equal to the present ones. Case PE_011 is not further considered, because the attribute «improvement goal» is different. Determining the similarity, case PE_007 is considered more similar than case PE_003, because the attribute «programming language» is equal to the actual one, in contrast to case PE_003 which is different.

The determination of similarity can be done by applying similarity measures [10]. Similarity measures determine for each case a numerical similarity value and induce a partial order among the cases in the case base. Global similarity measures are based on the number of attributes with corresponding values. In order to prevent that many cases may have a low similarity value, because only a few of the values are corresponding, although they might be quite similar, local similarity measures have to be applied.Local similarity measures also consider the similarity of values of attributes, For example, considering the attribute *«programming language»* specified as *C* in the present situation, cases with *«programming language: C++»* should also considered as potential reuse candidates, although the values are not equal, but similar. In this case, only its degree of similarity is lower than as the values would be equal. Reflecting the different degrees of importance of the individual characteristics for the determination of the similarity values, the given relevance factors have to be integrated into the similarity measure. In order to prevent overloading the user with less useful information, cases with a small degree of similarity are excluded by setting a threshold stating when cases are to be considered to be sufficiently similar.

Besides retrieving project-specific cases, also generic patterns can be retrieved. Generic patterns differentiate from project-specific cases by the level of abstraction of the values of context characteristics, e.g. instead of stating a particular programming language, as *Smalltalk*, they describe experiences related to object-oriented programming languages in general. Hence, if the user provides more abstract values as input to the retrieval, rather generic patterns are retrieved, than as if specific values are given, resulting in a set of project-specific reuse candidates. The different levels of abstraction of values are defined in the range definitions by taxonomies, which are used for the determination of the similarity [10].

Interactive support for reusing experienceware cases

The reuse candidates associated with the degree of similarity wrt. the given context characterization are presented to the user by means of a a knowledge navigation system. It allows the interactive exploration of the alternative case(s) by the user, facilitating the informed selection of the most adequate

PRESENT SITUATION

Context Characterization

organization	essential	Intellicar
department	less important	ABS
improvement goal	essential	improvement of reliability of sw system
staff size	irrelevant	100
programming language	less important	Ada
developer experience	important	high
GQM goal	important	unknown
measurement maturity	irrelevant	–

GQM-Experience Base (excerpt)

CASE PE 003

Context Characterization

organization	Intellicar
department	FI
improvement goal	improvement of reliability of sw system
staff size	150
prog. language	Fortran
dev. experience[a]	–
GQM goal	Analyze the *sw dev. process* in order to *characterize* wrt. *reliability* from the viewpoint of the *developers* at *Intellicar/FI.*
msmt maturity	low

a. Comment: Attribute was not captured in the past.

CASE PE 007

Context Characterization

organization	Intellicar
department	FI
improvement goal	improvement of reliability of sw system
staff size	150
prog. language	Ada
dev. experience	–
GQM goal	Analyze the *sw dev. process* in order to *characterize* wrt. *reliability* from the viewpoint of the *developers* at *Intellicar/FI.*
msmt maturity	low

CASE PE 011

Context Characterization

organization	Intellicar
department	FI
improvement goal	cost reduction in sw development
staff size	150
prog. language	Fortran
dev. experience	–
GQM goal	Analyze the *sw dev. process* in order to *characterize* the effort from the viewpoint of the *manager* at *Intellicar/FI.*
msmt maturity	medium

Figure 8. Simplified retrieval example

case, supporting the adaptation of the selected case to fit the current needs and, if necessary, the performance of a refined retrieval. This allows the usage of the GQM-EF without having an extensive domain knowledge and an understanding of the internal representation structure. Informed decisions are further supported by explicitly captured experiences about the reuses of a particular case in the past, concerning the costs of adaptations, occurred problems etc. (see Section 8.1.1.1). If the system fails to propose cases, general domain knowledge e.g. templates of GQM products are available to facilitate the planning of the measurement program from scratch.

8.3 Acquisition of Project-Specific Knowledge

The incremental evolution based on feedback from industrial applications is essential for continuous improvement of software engineering technologies. Concerning measurement programs, this can continuously reduce effort related to the planning phase and improve the consistency and effectiveness of measurement. Consequently, the knowledge in the GQM-EB has to be enhanced and updated each time a new measurement program is established. Each time GQM experienceware is reused to support the planning of a measurement program, newly gathered experiences in this measurement program are captured. To keep the effort related to the knowledge acquisition minimal, this process is intertwined with the reuse process: Informations provided by the user as input to the retrieval process and retrieved experiences from the GQM-EB are reused during the acquisition in order to optimize this process. While the user requests experience-based support, s/he provides a preliminary context characterization of the present situation in order to retrieve adequate reuse candidates from the GQM-EB. For example, while reusing experienceware in order to support the solution of a problem encountered (see Section 7) concerning the definition of a quality model for a question of the GQM plan «*What is the distribution of defects?*», the user provides the following context information: «organization: IntelliCar; application domain: automobile». In parallel to the retrieval process, this information is captured in a new case documenting current experiences. GQM-PEs reused for the development of GQM products in the present measurement program or GQM-PSEs reused for the solution of currently encountered problems are used to supplement the new case (see Figure 9). While capturing a new case, it has to be enhanced by basic informations (see Section 8.1), e.g., the acquisition technique used, which are asked to the user. The description of the new case has to be carefully reviewed by the user, still missing information has to be added and deviations from reused cases have to be modified, e.g., if a solution different to the one stated in the reused case was applied. The manual acquisition of new experiences is guided through the detailed case structure, which explicitly addresses relevant dimensions to be captured. The usage of organizational domain knowledge, such as glossaries and taxonomies, further facilitates the consistent description of experiences across software projects. If a case of the GQM-EB has been reused, its information on reuse has to be updated by obtaining the respective informations from the user. For example, considering the reuse of the GQM-PSE of Figure 4, concerning the «*dates of reuse*» the actual data is added and a comment on «*required adaptations: solution supplemented*» is appended.

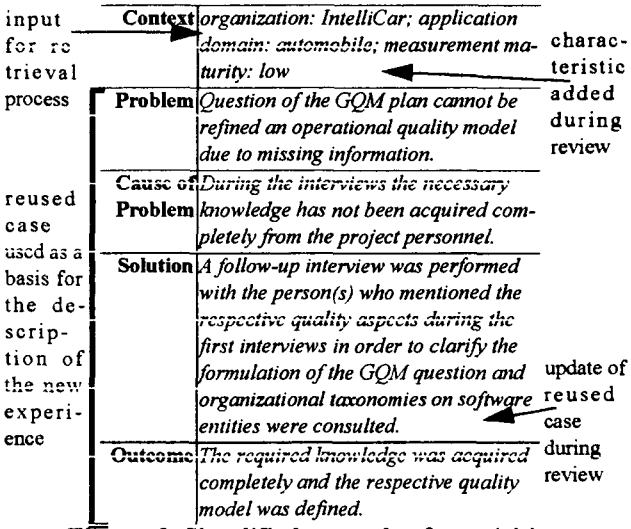


Figure 9. Simplified example of acquisition

The update and enhancement of general domain knowledge in the GQM-EB is also integrated into the reuse process. New entities used during the current measurement program can be

added by the user to the glossaries, taxonomies, and range definitions, enhancing general domain knowledge continuously.

8.4 Integration of new Experienceware

The new acquired project-specific cases and general domain knowledge have to be integrated into the existing experience base to be available for future reuse. The integration of new experienceware represents the learning process. This includes the selection of information to be captured, its appropriate representation in the GQM-EB and the identification of relevant context characteristics for indexing. This implies that cases have to be stored, interdependencies created or updated and, if necessary, generic patterns of cases have to be created or modified. In addition, general domain knowledge has to be updated and enhanced.

Project-specific cases (GQM-PEs or GQM-PSEs) are stored in the GQM-EB. Relationships between entities of a case are created. In addition, the project-specific cases are evaluated wrt. their similarity to other cases of the GQM-EB. If the new case differs only in small details from a case reused, an abstract case subsuming the two project-specific cases is created through case generalization [11]. The development of generic patterns through the knowledge engineer can be guided by related taxonomies which provide a basis for the derivation of abstractions subsuming the specific values.

The integration of new experienceware may also require the modification of the knowledge structure and indexing schemes of the experience base. Due to the fact, that the relevant context characteristics used as indices depend on the specific organization and may change over time, the continuous tailoring of the indexing scheme based on the relevance of context characteristics needs to be supported during the whole life cycle of a GQM-EB. For example, supplementary context characteristics of software projects may become relevant for the discrimination of cases. As shown in Figure 8, the attribute «*experience of developer*» had not been considered as a relevant characteristic for the context description of a case in the past, because all experiences were related to the same development team without variations in the level of experience. Since new measurement programs are established with other teams with different levels of experience, this attribute has been identified as important for the distinction of contexts and, consequently, has to be added to the context characterization. Continuous learning has also to take place wrt. the GQM-EF itself in order to improve and optimize its behavior. The performance of the GQM-EF regarding retrieval, acquisition and integration of knowledge has to be supervised, e.g. if and why retrieved reuse candidates are rejected by the user, and appropriately enhanced. This comprises the selection of indices and the adjustment of similarity measures in order to improve the retrieval results in the future.

9 Conclusion

For the successful application and continuous evolution of software engineering technologies in practice, experienceware has to be captured in corporate memories and reused in future applications. In this paper, we focus on the reuse of software measurement know-how in the planning of GQM-based measurement programs. There is a great potential for reducing the effort related to the planning phase and for improving the adequacy, consistency, and effectiveness of a measurement plan through reuse. We present a case-based reasoning approach for the operationalization of experience bases on measurement experienceware in industrial strength applications. Currently, we are implementing in cooperation with the Fraunhofer Institute for Experimental Software Engineering an experience base for the experience-based support of the planning of GQM measurement programs by using a CBR-tool [12]. The application of the approach offers the possibility of systematic acquisition of measurement experiences in practice and, therefore, provides a basis for further research on the evolution and generalization of technologies wrt. packaging and reuse of experienceware on measurement and software engineering technologies in general, based on feedback from its use in practice.

10 References

[1] V. R. Basili, G. Caldiera, H. D. Rombach. Experience Factory. In John J. Marciniak, ed., Encyclopedia of Software Engineering, vol. 1, pp. 528-532. John Wiley & Sons, 1994.

[2] L. C. Briand, C. M. Differding, H. D. Rombach. Practical Guidelines for Measurement-Based Process Improvement. Software Process Improvement and Practice, vol. 2, 1997.

[3] V. R. Basili, D. M. Weiss. A Methodology for Collecting Valid Software Engineering Data. IEEE Transactions on Software Engineering, SE-10(6):728-738, 1984.

[4] C. Gresse, L.C. Briand. Requirements for the Knowledge-Based Support of Software Engineering Measurement Plans. In Proceedings of the 9th Int. Conference on Software Engineering and Knowledge Engineering, Spain, 1997.

[5] J.R. Cho, R.C. Mathews. Interactions Between Mental Models Used in Categorization and Experiential Knowledge of Specific Cases. Journal of Experimental Psychology, 49A (3), 1996.

[6] K.D. Althoff, A. Birk, C. Gresse von Wangenheim, C. Tautz. CBR for Experimental Software Engineering. In H. D. Burkhard et al., eds., Case-Based Reasoning Technology from Foundations to Applications, Springer, to appear 1998.

[7] A. Aamodt, E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, vol. 17, nr. 1, 1994.

[8] C. Gresse, B. Hoisl, J. Wüst. A Process Model for GQM-Based Measurement. Technical Report STTI-95-04-E, Software Technology Transfer Initiative, Germany, 1995.

[9] C. Gresse von Wangenheim et al. Case-Based Reasoning Approach to Reuse of Experiential Knowledge in Software Measurement Programs. In Proceedings of 6th Workshop on Case-Based Reasoning, Germany, 1998.

[10] K. D. Althoff. Evaluating Case-Based Reasoning Systems: The Inreca Case Study. Habilitation, University of Kaiserslautern, Germany, 1996.

[11] R. Bergmann, W. Wilke. On the Role of Abstraction in Case-Based Reasoning. In Proceedings of European Workshop of Case-Based Reasoning, 1996.

[12] CBR-Works. TecInno GmbH, Germany (http://www.tecinno.de/tecinno_e/cbrwork.htm)

REMEX - A Case-Based Approach for Reuse of Software Measurement Experienceware

C. Gresse von Wangenheim. REMEX - A Case-Based Approach for Reuse of Software Measurement Experienceware. In Proceedings of the 3rd International Conference on Case-Based Reasoning, Germany, July 1999.

REMEX - A Case-Based Approach for Reusing Software Measurement Experienceware

Christiane Gresse von Wangenheim

Federal University of Santa Catarina - Production Engineering
Florianópolis, Brazil
gresse@eps.ufsc.br

Abstract. For the improvement of software quality and productivity, organizations need to systematically build up and reuse software engineering know-how, promoting organizational learning in software development. Therefore, an integrated support platform has to be developed for capturing, storing and retrieving software engineering knowledge. Technical support is complicated through specific characteristics of the software engineering domain, such as the lack of explicit domain models in practice and the diversity of environments. Applying Case-Based Reasoning, we propose an approach for the representation of relevant software engineering experiences, the goal-oriented and similarity-based retrieval tailorable to organization-specific characteristics and the continuous acquisition of new experiences. The approach is applied and validated in the context of the Goal/Question/Metric (GQM) approach, an innovative technology for software measurement.

Keywords. reuse, experience factory, case-based reasoning, software engineering, software measurement, GQM

11 Introduction

Today almost any business involves the development or use of software. However, state-of-the-practice is that software systems often lack quality and many software projects are behind schedule and out of budget [17]. In order to successfully plan, control and improve software projects, organizations need to continuously evolve software engineering (SE) know-how tailored to their specific characteristics and needs [8,10]. Experiences from their software projects have to systematically captured and reused across the organization. This enables the consolidation of organization wide SE know-how into competencies that empower the company to achieve considerable improvements and benefits [32]. Currently, reuse of SE knowledge is done in an ad-hoc, informal manner, usually limited to personal experiences. For the systematic acquisition and organization-wide communication of these experiences, corporate memories [2,8,18,20] have to be built (see Figure 29). In the software domain, the *Experience Factory* (EF) approach [8] proposes an organizational infrastructure for the analysis and synthesis of all kinds of software life cycle experiences or products. It acts as a repository for those and supplies these experiences to various software projects. However, for the operationalization of an EF in practice, we need a clever assistant that supplies the right experiences from the *Experience Base* (EB) to the user on demand.

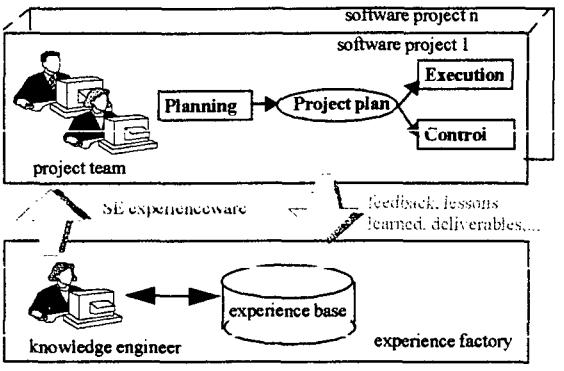


Fig. 29. Experience factory organization

In order to comprehensively support the software development process, various types of *experienceware* (EW) [18], including expertise and lessons learned (e.g., how to apply design inspections), quality models (e.g., distribution of rework effort per fault type), and deliverables (e.g., software measurement plans, requirement documents) related to several processes (e.g., design inspection, measurement) in different environments have to be retrieved addressing various purposes: facilitation of the planning or execution of software projects, prevention of past failures by anticipating problems, and guidance for the solution of occurring problems. And, since each software project is different, it is very unlikely to find an artifact fulfilling the needs of the actual project completely. Thus, experiences have to be retrieved from projects with "similar" characteristics, assuming that similar situations (or problems) require similar solutions. In the SE domain, for example, we assume, that measurement programs with similar goals use similar quality models or that similar problems occurring during design inspections have corresponding solutions. Due to the lack of general SE models in practice, organizational software know-how has to evolve in an incremental manner by learning from each new software project. Thus, the EF has to support continuous learning by capturing and integrating new experiences when software projects are planned and executed. In this context, *Case-Based Reasoning* (CBR) [5] plays a key role [10,18,20,27], as it provides a broad support for similarity-based retrieval for all kinds of EW and continuous incremental learning. However, the operationalization of the EF is not trivial, as relevant SE knowledge has to be identified, modeled and represented in the EB. Methods for goal-oriented retrieval providing support for different processes, objectives, and environments in the SE domain and for the continuous acquisition and integration of new experiences have to be developed. In this paper, we propose a case-based approach for an integrated support platform enabling organizational learning from SE experiences tailorable to organization specific characteristics. The approach is based on our experiences on reusing GQM-based measurement know-how (e.g., in the context of the industrial transfer and research projects [11,26]).

12 Reuse of GQM Measurement Plans

In this section, we give a short overview on software measurement, the application domain of our approach, and provide scenarios illustrating the reuse of measurement EW. Software measurement is an essential infrastructure technology for the planning, control and improvement of software projects. Organizations have to collect quantitative and qualitative data concerning their software products and processes, to build an em-

pirical justified body of knowledge. A specific technology for goal-oriented measurement is the *Goal/Question/Metric* approach (GQM) [9], which supports the definition and implementation of operationalizable software improvement goals. Based on a precisely specified measurement goal, relevant measures are derived in a top-down fashion via a set of questions and models. This refinement is documented in a GQM plan, providing a rationale for the selection of the underlying measures. Data is collected wrt. the measures and interpreted in a bottom-up fashion in the context of the models, question and goals, considering the limitations and assumptions underlying each measure. The establishment of measurement programs, which in practice requires a significant planning effort, can be substantially facilitated by reusing measurement EW [16], as illustrated in the following scenario.

Measurement Program at ABS/IntelliCar
GQM Goal
Analyze the software development process in order to improve the reliability from the viewpoint of the software developer at ABS/IntelliCar
GQM Questions
Q1. What is the total number of defects detected before delivery? Q2. What is the distribution of defects? Q3. Does the type of inspections have an impact on their effectiveness? Q4. Does the experience of developers have an impact on number of faults introduced in the system? ...
Quality Models
Effectiveness of inspections *Context*: company IntelliCar, automobile domain *Assumptions*: The defect density is comparable across documents. *Computation*: effectiveness = (number of defects detected in inspection)/(size of document * training duration) *Attributes*: number of defects detected in inspections; size of document; duration of training

Fig. 30. Excerpt of simplified example of GQM plan

Suppose a company, IntelliCar, which produces embedded software for automobiles has two main departments: FI which develops software for fuel injection devices and ABS which develops software for ABS brake control devices. As the company produces embedded software, one of its most important goals is to produce zero-defect software. Therefore, department FI established successfully a quality improvement program based on measurement two years ago. Now, also department ABS wants to start measurement-based improvement. As the contexts of both departments are similar and the improvement goal is the same, experiences available in department FI can be reused at ABS in order to reduce the planning effort and to improve the quality of the measurement program. Based on the measurement goal «*Analyze the software development process in order to improve the reliability from the viewpoint of the software developer at ABS/IntelliCar*», relevant quality aspects and influence factors have been acquired during interviews with the developers of department ABS. These are represented as a set of questions in the GQM plan, as shown in Figure 30. Now, in order to operationalize the questions of the GQM plan, quality models have to be developed. Assume, for example, that the question «*Q3. Does the type of inspection have an impact on the ef-*

fectiveness of inspections?», has also been evaluated in a similar measurement program in department FI. Then, the respective model can be reused, assessing its applicability based on its underlying assumptions. If necessary, the model is adapted to the specific characteristics of ABS. For example, assuming that inspector capabilities vary extensively between departments, the effectiveness of inspections is expected to depend not only on the size of the inspected document (as stated in the reused model), but also on the training of inspectors, then the new factor is included in the model.

While defining a model for question Q2, it turned out that an operational refinement of the question is impossible due to missing information concerning defect classification. The solution of this problem can be guided by experiences describing how a similar problem has been successfully solved at department FI (see Figure 31) by suggesting follow-up interviews in order to acquire the required information completely. In addition, reusing organizational glossaries can support the consistent usage of terms (e.g. *defect*) and reusing taxonomies representing generalization relations can help the refinement of abstract concepts (e.g. *«distribution of defects»* in Q2). Other phases of the measurement planning process can be supported accordingly through the reuse of measurement EW [16].

Context	company IntelliCar; department FI
Problem	Question of the GQM plan cannot be refined into an operational quality model due to missing information.
Cause of Problem	During the interviews the necessary knowledge has not been acquired completely from the project personnel.
Solution	A follow-up interview was performed with the person(s) who mentioned the respective quality aspects during the first interviews in order to clarify the formulation of the GQM question.
Outcome	The required knowledge was acquired completely and the respective quality model was defined.

Fig. 31. Example of problem experience

13 Representation of GQM Experienceware

In order to facilitate and improve the planning of GQM-based measurement programs through reuse of EW, an Experience Base is developed, modeling and representing relevant measurement EW.

13.1 GQM Experienceware Cases

As today wrt. most SE technologies no formal knowledge exists, the principal source are individual project experiences. Thus, SE EW is primarily captured in form of cases in the *GQM-Experience Base* (GQM-EB)[1], representing context-specific experiences gained in a particular software project in a specific organization. In order to provide comprehensive support, different types of EW cases are modeled by using a flexible, object-oriented frame-like representation formalism based on [24,28] and are stored in the GQM-EB [15,19]:

- *GQM Product Experienceware Case (GQM-PEC).* These cases include GQM products developed during the planning of a GQM-based measurement program. GQM-PECs are reused in similar software projects as a basis for the development of respec-

1.Here, we consider a specific instantiation of the experience base focusing on EW on the planning of GQM-based measurement programs.

tive products, resulting in a reduction of planning effort and improved quality of the GQM products.

- *GQM Problem-Solution Experienceware Case (GQM-PSEC).* GQM-PSECs explicitly capture problem solution strategies that have been adopted in past measurement programs (see Figure 31). Reusing GQM-PSECs can warn for potential failures in advance and guide a solution fitting the application context. Due to the specific nature of experiential knowledge, GQM-PSECs are represented as cases describing a specific problem, its cause, the solution applied and the outcome achieved.

Experienceware Cases are represented by a set of relevant attributes and interdependencies based on domain models (see Section 13.2) [29]. To enable the retrieval of EW cases from similar software projects, the environment from which the case has been obtained is characterized. This is done through a minimal set of characteristics (e.g., business sector, improvement goals, development process used), which allows to identify similar cases and to discriminate different ones. In order to assess the reuse potential of the case, cases are enhanced by basic information (e.g., viewpoint, representativeness). Information about past reuses of a case, such as preconditions for reuse, required adaptations, cost and frequency of reuse, are explicitly captured [19] in order to facilitate the reuse of experiences and their adaptation to specific project characteristics.

13.2 General Domain Knowledge

In order to model relevant EW and facilitate the consistent representation and acquisition of new experiences across software projects, general domain knowledge on GQM EW is represented in the GQM-EB [16,19].

GQM EW Models. Entities related to GQM EW are explicitly modeled in a hierarchy of classes [16,28] (see Figure 32). Each class is structured by a set of attributes repre-

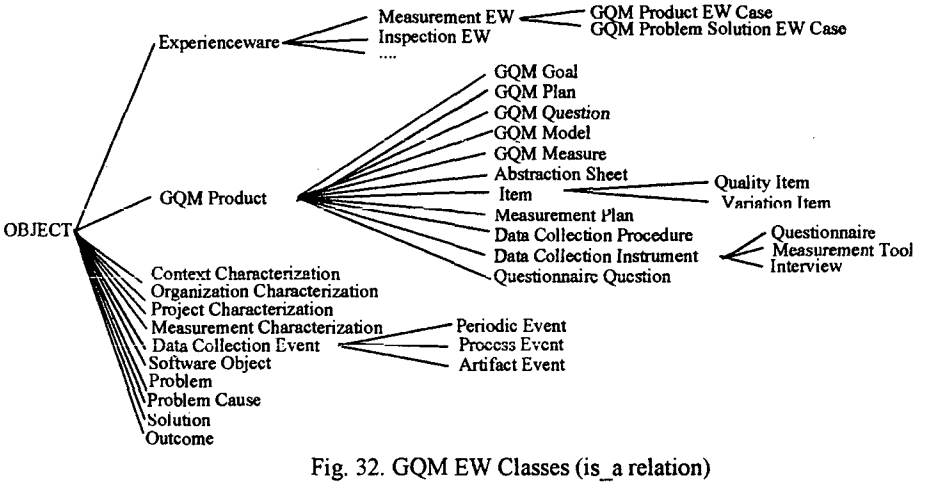


Fig. 32. GQM EW Classes (is_a relation)

senting basic values or relationships to other entities. Attributes are defined through an identifier, description, cardinality, its type or kind of relationship, a default value and explicitly stating if the attribute has to be specified (mandatory) when a new instance of this class is acquired [15].

Class	GQM Measure					
Description	defines data to be collected					
Attributes	**Identifier**	**Description**	**Cardinality**	**Type or Kind**	**Default**	**Mandatory**
	id	identifies the GQM measure	1	Identifier	-	yes
	assumptions	about the applicability of the measure	0..1	Text	none	no
	description	describes data to be collected	0..1	Text	-	yes
	scale	defines scale of the measure	0..1	Scale	-	yes
	unit	declares unit of the measure	0..1	Unit	-	no
	range	declares range of the values of the measures	0..1	Text	-	no
	model	references the corresponding model	0..*	defined-by [GQM-Model])	-	yes
	...					

A GQM measure defines which data has to be collected. It includes the explicit definition of assumptions concerning the application of the measure. Regarding the expected values, scale, unit (only in case of numerical values) and range have to be defined.As GQM measures are derived from models which determine the attributes to be measured, this is represented as a defined-by relation. Based on the GQM measure the collection procedure defining when, how, and by whom the data has to be collected is defined.

Fig. 33. Simplified example of the class GQM Measure

Type Definitions. Type definitions model qualities of SE entities, such as, developer experience, or categorize concepts, e.g., programming languages as Unordered Symbol with the possible values «Delphi, C++, etc.». Type definitions are used to type class attributes. They facilitate the situation assessment and support the manual adaptation of retrieved EW cases by explicitly indicating alternatives, as well, as the consistent acquisition of experiences across projects. For each type, its related supertype, range and the local similarity measure are specified. For example, the experience level of developers might be classified through the Ordered Symbols: none, low, medium, high, using the standard local similarity measure for ordered symbols. For symbol types, the meaning of each value is explicitly defined through range definitions, e.g., «high» experience may be defined as worked for more than 2 years in the application domain. In addition, for numerical types, the unit is explicitly stated, e.g., person-hours.

Glossaries. Glossaries define terminology and basic concepts related to software measurement [16,19]. For example, «Failure: is the inability of the software to perform a required function wrt. its specifications». A glossary supports the adequate use of terms, their consistency across an organization, and ensures that the reuse of GQM products is based on sound assumptions.

Taxonomies. Taxonomies represent ordered arrangements of entities according to their presumed relationships, e.g., organization hierarchy [16,19]. They guide the appropriate refinement of objects of interest during the situation assessment and acquisition of new experiences.

13.3 Knowledge Levels

Software products, processes, resources as well as characteristics and terminology vary between different organizations. Therefore, the domain model has to be tailored to the specific environment. Generally, we can identify different levels of knowledge valid in different scopes of domains:

- **Software measurement domain.** Here, general knowledge on GQM-based measurement is represented, which is transferrable between organizations. This level includes

GQM EW models, general valid types and range definitions (e.g., on measurement scale), and general valid terms in the glossary (e.g., software process).

- **Organization domain.** Here, organization specific knowledge related to software measurement is represented. If the GQM technology is modified in a particular organization, the respective knowledge from the upper level is adapted accordingly. Type and range definitions are enhanced by organization specific definitions. For example, one organization could classify «*experience of the developers*» into the categories *(expert-participated in system development; medium-participated in training; none)*, whereas another organization might classify experience into (*high-working for more than 2 years, medium-worked once, low-never worked in application domain*). The glossary and taxonomics are completed by organization specific terms.
- **Project domain.** At this level, instantiations of GQM EW cases are represented gathered from particular software projects. For example, a GQM-PEC including a GQM plan from a measurement program of the project HYPER at the department ABS/IntelliCar.

14 Experience-Based Support of GQM Planning

14.1 Determination of Retrieval Goals

During the planning of GQM measurement programs the GQM-EB can be inquired to find useful EW to guide, support and improve various SE tasks in a specific environment. In order to provide comprehensive support for several SE tasks, various types of experiences have to be retrieved, from different viewpoints in different environments addressing various purposes: support of software projects by reusing similar products developed in the past, prevention of failures by anticipating problems, guidance for the solution of problems by reusing solution strategies adopted in past similar problems, and the identification of patterns of experiences for the maintenance and evolution of the EB. Thus, a goal-oriented retrieval method [14] is developed that retrieves a set of relevant experiences wrt. a specific reuse goal. Based on reuse scenarios, retrieval goals are determined explicitly specifying the following dimensions:

Retrieve <object>
to <purpose>
concerning <process>
from the <viewpoint>
in the context of <environment>

For example, «*retrieve lessons learned to guide problem solution concerning software measurement from the viewpoint of quality assurance personnel at IntelliCar*».

Based on the retrieval goals, reusability factors are determined. This includes the specification of relevant indexes[1] and their importance and the parametrization of the similarity measure. For example, for the retrieval of a solution strategy, relevant indexes

1. As index we denote attributes of the case, which predict the usefulness of the case concerning the given situation description, and which are used for retrieval and determination of the similarity value.

might be the problem description and the task when the problem occurred, whereas potential problems wrt. a specific task might be identified based on the task only.

14.2 Retrieval Process

Considering different retrieval goals, a goal-oriented method for similarity based retrieval is defined, including the following steps [14]:

Step 5. Situation assessment. The current situation is described by the user specifying the retrieval goal and a set of indexes related to the specific retrieval goal based on a predefined indexing scheme. The importance of each index wrt. the specific retrieval goal is stated through a *relevance factor* assigned to each index. Relevance factors are stored in the EB and can be manually adapted by the user. To facilitate the assignment, relevance factors are classified into «essential, important, less important, irrelevant». Indexes marked as essential are perfectly matched to the ones in the situation assessment, the ones marked as important or less important are partially matched and the ones marked as irrelevant are not further considered. Unknown indexes are explicitly marked. Table 16 illustrates a situation assessment with an exemplary set of indexes. The situation assessment is further supported by general domain knowledge [19]: glossaries can be used for a consistent usage of terminology across projects and taxonomies guide and direct the appropriate definition of indexes. Type and range definitions facilitate the identification of the present values and guarantee a consistent description across projects.

Reuse goal			GQM Experience Base (excerpt)		
object	lesson learned (PSEC)		**CASE PSEC_003**	**CASE PSEC_007**	**CASE PSEC_011**
purpose	guide problem solution				
process	sw measurement				
viewpoint	quality assurance personnel				
environment	IntelliCar				
Indexes					
department	irrelevant	ABS	Fuel Injection	Fuel Injection	Fuel Injection
staff size	less important	10	15	100	50
application domain	essential	automobile	automobile	automobile	automobile
improvement goal	important	improvement of sw system reliability	improvement of sw system reliability	improvement of sw system reliability	cost reduction in sw development
programming language	irrelevant	Ada	Fortran	Ada	C
dev. experience	less important	high	medium	low	low
sw system size	less important	*unknown*	15 KLOC	80 KLOC	60 KLOC
measurement maturity	important	initial	--	--	--
task	essential	measurement goal definition	measurement goal definition	development of measurement plan	measurement goal definition

Table 16. Simplified retrieval example

Step 6. Exact matching of indexes marked as essential. In a first step, the cases of the EB are perfectly matched with the situation assessment wrt. the indexes marked as essential, determining a set of potential reuse candidates. Table 16 shows a simplified example: while comparing the cases of the EB with the situation assessment, case PSEC_03 and PSEC_11 are considered as potential reuse candidates, because the values of the indexes marked as essential («application domain» and «task») are equal to the present ones. PSEC_07, which describes an experience regarding the develop-

ment of the measurement plan, is not further considered, as the value of the index «task» is different to the one of interest.

Step 7. Partial matching of similar cases. For all potential reuse candidates a similarity value is computed by partially matching the indexes (except the ones marked as *essential)* using a specific similarity measure wrt. the retrieval goal (see Section 14.3). Cases with a higher similarity value than a given threshold are considered as sufficiently similar and proposed to the user as reuse candidates ranked by their similarity values. Continuing the example shown in Table 16, case PSEC_03 is considered more similar to the given situation than PSEC_11, because the values of the indexes of PSEC_03 marked as important or less important are more similar to the current ones (especially regarding «staff size», «improvement goal»).

Step 8. Selection of reuse candidate(s). Based on the proposed reuse candidates the user can select the most appropriate case(s) and, if necessary, manually adapt them to fit the current needs. Informed decisions are further supported by experiences explicitly captured in the EB about the reuses of a particular case in the past [19] (see Section 13.1). If the system fails to propose reuse candidates, general domain knowledge, e.g., GQM product models, is available to support the SE tasks.

14.3 Similarity Measure for the Retrieval of Experienceware

For the identification of «similar» EW cases concerning various retrieval goals (see step 3/Section 14.2), we define a generic similarity measure sim(Sit',E_k') [14] that can be parameterized for a specific goal. Taking into account specific characteristics of the SE domain, such as the lack of explicit domain models in practice, diversity of environments, incompleteness of data, and the consideration of «similarity» of experiences, the similarity measure is based on the following assumptions (see [14] for details):

- Depending on the retrieval goal, a particular set of indexes is defined for situation assessment and matching. A set of indexes C is represented as a list of features $C_g=\{C_{g1}, C_{g2},...\}$ wrt. the particular retrieval goal g. The range of the value c_i of the feature C_{gi} is defined by the respective range definition W_i (see Figure 34).

Example Index Set		Type/Range	Relevance Vector R_g	Present Situation Sit'={(C_{gi}, s_i)		Case E_k	FS
C_1	staff size	Interval of numbers [0,50]	less important (0.15)	s_1	10	15	W
C_2	improvement goal	String	important (0.35)	s_2	"improvement of system reliability"	"improvement of system reliability"	E
C_3	measurement maturity	Ordered Symbol: {initial, low, routine}	important (0.35)	s_3	initial	*unknown*	U
C_4	sw system size	Number [0.100]	less important (0.15)	s_4	*unknown*	15KLOC	R

Fig. 34. Example

- The present situation is assessed based on the set of indexes wrt. the retrieval goal, represented as a list of feature-value pairs Sit'={(C_{gi}, s_i) ∈ Sit | relevance factor (S_i) ≠ essential} including the features $C_{gi} \in C_g$ and their values $s_i \in W_i$.
- In the EB, an EW $case_k = (E_k,\varepsilon_k)$ represents an experience by feature-value[1] pairs (experience $E_k=\{(E_{k1}, e_{k1}), (E_{k2}, e_{k2}),...\}$ with the features E_{ki} and their values $e_{ki} \in W_i$

(and with $E_k' \subseteq E_k$ and $\forall\ E_{ki}' \in C_g$ and their respective values e_{ki}'), describing the know-how gathered in a software project, the context from which its originates, and its relationships (see Figure 34). In addition, a threshold $\varepsilon_k \in [0,1]$ is stated for each case that determines, if the case is sufficiently similar to the situation assessment to be proposed as a reuse candidate.

- In the SE domain, many cases may have a low similarity value, due to few identical values, although they might be quite similar (e.g. programming languages C and C++). Thus, local similarity measures are introduced. Generic local similarity measures $\upsilon'(s_i,e_{ki}') \in [0,1]$ for basic value types W(v) are defined in [19,28]. Local similarity thresholds $\theta_i \in [0,1]$ are introduced for each index C_{gi} determining if the values are considered as (sufficiently) similar.
- *Relevance factors* are defined, which reflect the importance of a feature concerning the similarity of cases wrt. a specific retrieval goal (see Figure 34). Here, for each retrieval goal *g* a specific index set C_g is used. Thus, for each index $C_{gi} \in$ index set C_g, a relevance factor $\omega_{gi} \in [0,1]$ is defined in dependence on the specific retrieval goal g. For each retrieval goal, those relevance factors are represented by a *relevance vector* $Rg = \{\omega g_1, \omega g_2, \ldots\}$ with $\sum \omega g_i = 1$ normalized in the EB.
- In order to explicitly deal with incomplete knowledge, the similarity of two objects is expressed through a linear contrast of weighted differences between their common and different features [6,30]. The following *Feature Sets* (FS) are distinguished:
  - E: Set of corresponding features of the given situation and the stored case ($E = \{Cg_i \mid (Cg_i \in Sit' \cap E_{ki}')$ and $(\upsilon'(s_i,e_{ki}') \geq \theta_i)\}$). For example, if both, the situation assessment and the stored case state the feature «experience of developer» as high.
  - W: Set of contradicting features of the given situation and the stored case ($W = \{Cg_i \mid (Cg_i \in Sit' \cap E_{ki}')$ and $(\upsilon'(s_i,e_{ki}') < \theta_i)\}$). For example, if in the past no effort reporting tools were available, but now in the given situation the feature «effort reporting tools» is stated as available.
  - U: Set of unknown features in the actual situation description ($U = \{Cg_i \mid (Cg_i \in E_{ki}' - Sit')\}$). For example, when initiating a software project certain information, such as «software system size» may be stated as unknown in the situation description.
  - R: Set of redundant features not contained in the stored case ($R = \{Cg_i \mid (Cg_i \in Sit' - E_{ki}')\}$). For example, the feature «developer experience» may not have been considered initially, but later become important for the identification of relevant cases.

  For each set, a specific weight $\alpha, \beta, \gamma, \delta \in [0,1]$ is defined.

The global similarity measure is defined as:

$$sim(Sit',E_k') = (\alpha \sum_{si \in E} \omega_{ik} \upsilon'(s_i,e_{ki}')) / ((\alpha \sum_{si \in E} \omega_{ik} \upsilon'(s_i,e_{ki}')) + (\beta \sum_{si \in W} \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))) + (\gamma \sum_{si \in U} \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))) + (\delta \sum_{si \in R} \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))))$$

Based on the similarity value calculated, a $case_k$ is considered as reuse candidate, if all features marked as essential in the given situation exactly match the respective features

1.Here, values represent atomic values or relations to other entities.

of the case, and if sim(Sit',Ek') ≥ global similarity threshold ε_k of $case_k$.

15 Continuous Acquisition and Integration of Experienceware

The incremental evolution based on feedback from industrial applications is essential for continuously building and improving SE know-how. Consequently, the knowledge in the GQM-EB has to be enhanced and updated each time a new measurement program is run in the organization. This means that we have to continuously capture new experiences from the quality assurance personnel. In order to keep the effort related to the knowledge acquisition minimal, this process is intertwined in the retrieval/reuse process (see Figure 35): Information provided by the user as input to the retrieval process, such as a context characterization, and reused experienceware from the GQM-EB are in parallel used for the creation of new EW cases.

For example, while reusing EW in order to support the solution of a problem encountered (see Section 12) concerning the definition of a quality model, the user provides the following situation assessment: «*organization:* InteliiCar; *application domain:* automobile; *problem:* Question of GQM plan cannot be refined into model». This information is used for the retrieval process, and in parallel for the description of a new case documenting experiences regarding the present situation. Information contained in a similar case retrieved and reused in order to solve the current problem, is used to supplement the new case description (see Figure 36). The generated new case is reviewed by the user before storage in the EB. Additional information (e.g. basic information) is added, and if necessary, deviations from the reused case are adjusted, e.g., if a solution different to the one stated in the reused case was applied. The acquisition of new experiences is further guided through GQM EW models, which explicitly address relevant dimensions to be captured. Glossaries and taxonomies facilitate the consistent description of experiences across software projects. The new acquired experiences are integrated into the existing EB to be available for future reuse. This implies that EW cases have to be stored, domain models enhanced and, if necessary, generic patterns of cases have to be created or modified.

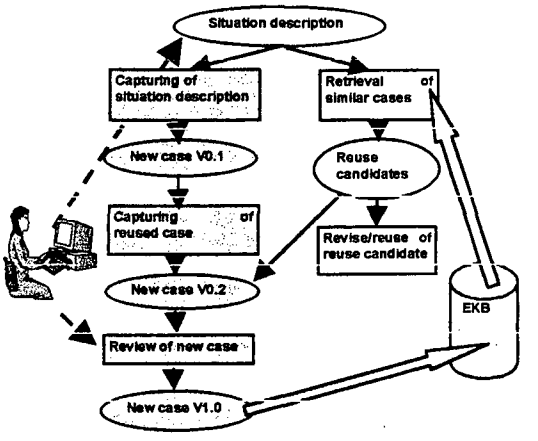


Fig. 35 Integration of acquisition process

Project-specific cases (GQM-PECs or GQM-PSECs) acquired in parallel to the retrieval process are stored as instances of GQM EW cases in the GQM-EB. Based on protocols on the retrieval/reuse process and comparisons of the reused and new case, reuse information is added to the reused case. For example, the date of reuse is added, the frequency of reuse is increased, and attributes which have been adapted to fit the new situation are explicitly listed.

In addition, project-specific cases are evaluated wrt. their similarity to other cases of the GQM-EB. If the new case differs only in small details from a case reused, an abstract case subsuming the project-specific cases is created through case generalization. The development of generic patterns through the knowledge engineer can be guided by taxonomies which provide a basis for the derivation of abstractions.

Based on an evaluation of new terms defined in the specific GQM EW case through the knowledge engineer, the organizational glossary and taxonomies are enhanced.

The continuous evolution and customizing of the EF to a specific environment may also require the modification of the representation of EW cases, the indexing scheme and similarity measure based on user feedback from the application. Due to the fact, that indexes depend on the specific environment and may change over time, the continuous tailoring of the indexing scheme needs to be supported during the whole life cycle of a GQM-EB through the knowledge engineer. For example, supplemental context characteristics of software projects may become relevant for the discrimination of cases. As shown in Figure 36, the attribute «*measurement maturity*» had not been considered as a relevant characteristic for the context description of a case in the past, because all experiences were related to projects without variations concerning the maturity. Since a new measurement program is established in a project with a different level of maturity, this attribute has become relevant for the distinction of cases and is added to the context characterization. Continuous learning has also to take place wrt. the similarity measure and its parametrization for specific retrieval goals in order to improve and optimize its performance. Therefore, the retrieval and reuse process is supervised and, based on the feedback, appropriately tailored to the specific environment through the knowledge engineer. Here, protocols documenting the user's (re-)actions and user-provided critics and suggestions can serve as a basis for the maintenance through the knowledge engineer (see Table 17). Based on a careful analysis of the causes, the selection of indexes and/or the

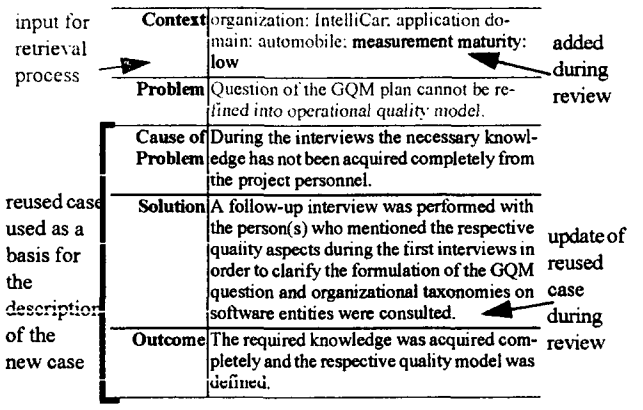


Fig. 36. Simplified example of acquisition

Feedback	Implication for update
Index manually added for retrieval	•Addition of index to the indexing scheme
Relevance factor manually modified	•Modification of weight assigned to the index •Index frequently marked as irrelevant might be removed from the index scheme

Table 17. Examples of retrieval feedback and its implications

Increasing number of retrieved reuse candidates	•Changing optimistic strategy for similarity measure into a more pessimistic •Increase of tresholds
Frequent rejection of cases suggested as reuse candidates	•If a specific case is affected: increase of global treshold of the case •If different cases are affected: review of indexing scheme and similarity measure under consideration of additional critics and suggestions of the user

Table 17. Examples of retrieval feedback and its implications

similarity measure have to be adapted accordingly in order to improve retrieval results in the future.

16 Discussion

In the software domain, various approaches exist for reuse primarily focusing on software code, e.g, based on library and information science, knowledge-based systems, or database management technologies [12]. However, the majority of those approaches fails to recognize the complexity of SE experience in general, often requires a thorough classification of the domain, or does not provide any means for similarity-based retrieval.

Recently, CBR has been recognized as a promising approach for the operationalization of learning organizations in the SE domain [2,3,18,22,27]. Applications are developed in different SE areas, like capturing and formalizing best practices (e.g., [20]), effort prediction (e.g., [13]), change management [23], and requirements acquisition (e.g., [25]). However, so far there does not exist an approach on reusing software measurement EW. Only few approaches offer flexible similarity-based retrieval methods, for example, through a context concept as a "similarity environment for the retrieval" [1,31], dynamic ranking of importance ratings of indexes [21], or partitioning the case base through the use of *prototypes* [7]. However, if multiple retrieval goals have to be supported by a case base, this is not sufficient. The creation of distinct case bases for test selection and diagnosis in PATDEX [6,31], can be seen in analogy to different retrieval goals, although inefficient due to administration and maintenance reasons. In contrast, our approach, systematizes the concept of goal-oriented retrieval through a flexible and tailorable retrieval method and similarity measure based on the advanced similarity model of PATDEX which explicitly deals with unknown information, filter attributes, and local similarity measures.

Besides integrating experiential knowledge (in form of cases) and general domain knowledge as in several CBR systems, our approach explicitly models different levels of knowledge focusing on different scopes.

Concerning the tailoring and continuous evolution of the EF to organization specific characteristics, only a few systems offer mechanisms for the systematic and integrated acquisition of user feedback and learning possibilities regarding the similarity measure as, e.g., the tailoring of relevance factors (see [4] for an overview), which represent the basis for the continuous evolution of our approach.

17 Conclusion

For the successful planning and improvement of software measurement, EW has to be

captured in corporate memories and reused across the organization. Based on our experiences on the application of the GQM approach in practice, we develop a case-based approach for the operationalization of organizational learning in software measurement focusing on the technical aspects. Relevant measurement EW is modeled, a goal-oriented method for similarity-based retrieval tailorable to specific environments is developed, and an acquisition process intertwined into the retrieval/reuse process described. Currently, we are implementing the approach. Further empirical research will have to be carried out in experiments and industrial transfer projects to assess strengths and weaknesses of the approach.

References

1. Althoff, K.-D., et al.: Case-Based Reasoning for Decision Support and Diagnostic Problem Solving: The INRECA Approach. Proc. 3rd German Workshop on Case-Based Reasoning, Germany (1995)
2. Althoff, K.-D., Bomarius, F., Tautz, C.: Using Case-Based Reasoning Technology to Build Learning Software Organizations. Proc. of Workshop on Building, Maintaining, and Using Organizational Memories at the 13th European Conference on AI (1998)
3. Althoff, K.-D., et al.: CBR for Experimental Software Engineering. In M. Lenz et al. (eds.), Case-Based Reasoning Technology - From Foundations to Applications, LNAI 1400, Springer Verlag (1998)
4. Althoff, K.-D.: Evaluating Case-Based Reasoning Systems: The Inreca Case Study. Postdoctoral Thesis, University of Kaiserslautern, Germany (1997)
5. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 17(1) (1994)
6. Althoff, K.-D., Wess, S.: Case-based Knowledge Acquisition, Learning and Problem Solving in Diagnostic Real World Tasks. Proc. of the 5th European Knowledge Acquisition for Knowledge-Based Systems Workshop, Scotland/UK (1991)
7. Barletta, R.: A Hybrid Indexing and Retrieval Strategy for Advisory CBR Systems Built with ReMind. Proc. of the 2nd European Workshop on Case-Based Reasoning (1994)
8. Basili, V. R., Caldiera, G., Rombach, H. D.: Experience Factory. In J. J. Marciniak (ed.), Encyclopedia of Software Engineering, John Wiley & Sons (1994)
9. Basili, V. R., Caldiera, G., Rombach, H. D.: Goal Question Metric Paradigm. In J. J. Marciniak (ed.), Encyclopedia of Software Engineering, John Wiley & Sons (1994)
10. Barr, J.M., Magaldi, R.V.: Corporate Knowledge Management for the Millennium. In I. Smith, B. Faltings (eds.), Advances in Case-Based Reasoning, Springer Verlag (1996)
11. CEMP Consortium. Customized Establishment of Measurement Programs. Final Report, ESSI Project Nr.10358 (1996)
12. Frakes, W. B., Gandel, P. B.: Representing Reusable Software. Information and Software Technology, 32(10) (1990)
13. Finnie, G. R., Wittig, G. W., Desharnais, J.-M.: Estimating Software Development Effort with Case-Based Reasoning. Proc. of the 2nd Int. Conf. on Case-Based Reasoning, RI (1997)

14. Gresse von Wangenheim, C., Althoff, K.-D., Barcia, R.M.: Intelligent Retrieval of Software Engineering Experienceware. Proc. of the 11th Int. Conf. on Software Engineering and Knowledge Engineering, Germany (1999)
15. Gresse von Wangenheim, C.: REMEX - A Case-Based Approach for Reuse of Software Measurement Experienceware. Technical Report PPGEP-C3002.99E, Graduate Program in Production Engineering, Federal University of Santa Catarina, Brazil (1999)
16. Gresse, C., Briand, L. C.: Requirements for the Knowledge-Based Support of Software Engineering Measurement Plans. Journal of Knowledge-Based Systems, 11 (1998)
17. Gibbs, W.W.: Software's Chronic Crisis. Scientific American (1994)
18. Gresse von Wangenheim, C.: Knowledge Management in Experimental Software Engineering - Create, Renew, Build and Organize Knowledge Assets. Proc. of the 10th Int. Conf. on Software Engineering and Knowledge Engineering, San Francisco, California (1998)
19. Gresse von Wangenheim, C., von Wangenheim, A., Barcia, R. M.: Case-Based Reuse of Software Engineering Measurement Plans. Proc. of the 10th Int. Conf. on Software Engineering and Knowledge Engineering, San Francisco, California (1998)
20. Henninger, S.: Capturing and Formalizing Best Practices in a Software Development Organization. Proc. of the 9th Int. Conf. on Software Engineering and Knowledge Engineering, Spain (1997)
21. Kolodner, J. L.: Case-Based Reasoning. Morgan Kaufmann, San Francisco, California (1993)
22. Kitano, H., Shimazu, H.: The Experience-Sharing Architecture. In D. Leake (ed.), Case-Based Reasoning Experiences: Lessons Learned & Future Directions (1996)
23. Lam, W., Shankararaman, V.: Managing Change During Software Development: An Incremental, Knowledge-Based Approach. Proc. of the 10th Int. Conf. on Software Engineering and Knowledge Engineering, San Francisco, California (1998)
24. Manago, M. et al.: Casuel: A Common Case Representation Language. Technical Report Deliverable D1, Esprit Project Inreca P6322 (1994)
25. Maiden, N.A., Sutcliffe, A. G.: Exploiting Reusable Specifications Through Analogy. Communications of the ACM, 35(4) (1992)
26. Kempter, H., Leippert, F.: Systematische Software-Qualitätsverbesserung durch zielorientiertes Messen und Bewerten sowie explizite Wiederverwendung des Software-Entwicklungs-Know-how. Proc. of the BMBF-Seminar Software Technology, Germany (1996)
27. Tautz, C., Althoff, K.-D.: Using Case-based Reasoning for Reusing Software Knowledge. Proc. of the 2nd Int. Conference on Case-Based Reasoning, Springer Verlag (1997)
28. Tautz, C., Gresse von Wangenheim, C.: REFSENO: A Representation Formalism for Software Engineering Ontologies. Proc. 5th German Conf. on Knowledge-Based Systems, Germany (1999).
29. Tautz, C., Gresse von Wangenheim, C.: REFSENO: A Representation Formalism for Software Engineering Ontologies. Technical IESE-Report 015.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany (1998).

30. Tversky, A.: Features of Similarity. Psychological Review, 84 (1977)
31. Wess, S.: Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik. Ph.D. Thesis, University of Kaiserslautern, Germany, infix Verlag (1995)
32. Zand, M., Samadzadeh, M.: Software Reuse: Current Status and Trends. Journal of Systems and Software, 30 (3) (1995)

Intelligent Retrieval of Software Engineering Experienceware

C. Gresse von Wangenheim, K.-D. Althoff, R. M. Barcia. Intelligent Retrieval of Software Engineering Experienceware. In Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering, Germany, June 1999.

Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering, Germany, June 1999

Intelligent Retrieval of Software Engineering Experienceware

Christiane Gresse von Wangenheim
Federal University of Santa Catarina
Florianópolis, Brazil
gresse@eps.ufsc.br

Klaus-Dieter Althoff
Fraunhofer Institute IESE
Kaiserslautern, Germany
althoff@iese.fhg.de

Ricardo M. Barcia
Federal University of Santa Catarina
Florianópolis, Brazil
rbarcia@eps.ufsc.br

Abstract

For the continuous improvement of quality and productivity in software development, organizations have to build up a specific body of software engineering know-how. Software know-how has to be systematically collected from projects, stored in a corporate memory, and shared across the organization. To support and guide software projects, useful experiences have to be identified and retrieved from the knowledge base. As support is required for different processes, purposes, and environments, the usefulness of retrieved experiences depends mainly on the particular reuse situation. Thus, a flexible retrieval method and similarity measure is required, which can be continuously tailored to the specific situation based on feedback from its application in practice. This paper proposes a case-based approach for the retrieval of software engineering experienceware taking into account those specific characteristics of the software engineering domain, such as the lack of explicit domain models in practice, diversity of environments and software processes to be supported, incompleteness of data, and the consideration of «similarity» of experiences.

Keywords: case-based reasoning, retrieval, similarity measure, software engineering experienceware, software process improvement.

1. Introduction

Today software engineering (SE) processes in practice are frequently of insufficient quality, productivity, and predictability. Many organizations have established their software processes in an ad hoc manner, which has often led to incompatible solutions and negative consequences. Essential for the continuous improvement of software quality and productivity is a company's specific software know-how [12]. Companies have to gather experience from their software projects and share them across the organization. This includes several kinds of experience, denoted as experienceware [19], for example, expertise and lessons learned wrt. SE technologies (e.g., on how to apply design inspections), quality models (e.g., distribution of rework effort per fault type), and deliverables (e.g., software measurement plans, requirement documents). In order to systematically collect and reuse software engineering know-how tailored to the specific characteristics and needs of a software organization, Experience Factories (EF) [3,10,19,21] have to be built (see Figure 1). For their operationalization in practice, we need a clever assistant that supplies the right experienceware from the Experience Base (EB) to the user on demand, learns from its own experience, and continuously adapts to the specific environment. Essential for the usage and acceptance of an EF in practice is the usefulness of the SE experienceware retrieved from the EB to support new software projects. As the usefulness of experiences can only be determined when tried to be reused in the current situation, the usefulness is predicted through the criterion of similarity of the situation described in the experience and the current one, assuming that similar situations (or problems) require similar solutions. In the software engineering domain, for example, we assume, that measurement programs with similar goals use similar quality models or that similar problems occurring during design inspections have corresponding solutions.

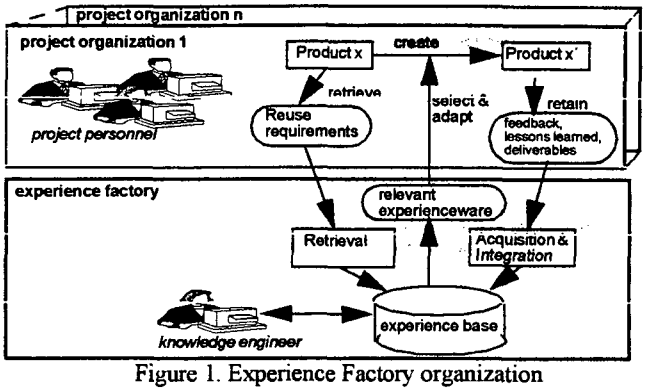


Figure 1. Experience Factory organization

In this context, Case-Based Reasoning (CBR) [6] plays a key role [12,19,26], as it provides a broad support for similarity-based retrieval for all kinds of experienceware and continuous incremental learning. Reuse candidates in a CBR system are retrieved by partially matching the given situation description with the cases in the case base and inducing a total order among the cases based on numerical similarity values assigned to each case through a similarity measure [4,25,30].

However, the definition of a retrieval method and similarity measure for the retrieval of SE experienceware is not trivial as support for different objects, processes, purposes, view-

points, and environments is required. For example, experienceware cases can be (re-)used during several software processes, such as design inspections or measurement programs, for various purposes: facilitate the planning and execution of software processes or guide the solving of problems, from different viewpoints, e.g., quality assurance personnel or knowledge engineer. And, what represents relevant software know-how differs among companies wrt. their context, needs, and business goals. This requires a flexible retrieval method and similarity measure that can be tailored to different reuse goals.

Some CBR approaches enable the flexibilization of the similarity measure for a specific application or class of application. Based on experiences on reusing software measurement experienceware in industrial projects (e.g., in the context of the project CEMP [14]), we propose a flexible approach that systematizes those CBR approaches for the goal-oriented retrieval of SE experienceware. Based on the similarity model applied in PATDEX [7], our approach deals explicitly and in a tailorable way with different retrieval goals as well as incompleteness of information and enables a comprehensive consideration of global and local similarity.

2. Requirements for the Retrieval of Similar Software Engineering Experienceware

In this chapter, important requirements for the effective and efficient retrieval of SE experienceware are identified. The list of requirements is based on specific requirements for experience-based support of planning of measurement programs [18,19] derived from our experiences applying measurement in industry [14] and from other software engineering areas [3,13,10,21]:

Goal-Oriented Retrieval. In order to provide comprehensive support for several SE tasks, various types of experiences have to be retrieved, for diverse tasks, from different viewpoints, in different environments addressing various purposes: support of software projects by reusing similar products developed in the past, prevention of failures by anticipating problems, guidance for the solving of problems by reusing solution strategies adopted in past similar problems, and the identification of patterns of experiences for the maintenance and evolution of the EB. Thus, a goal-oriented retrieval method has to be defined that retrieves a set of relevant experiences for the specific reuse goal. Various parameters of the retrieval process have to be adapted to the specific reuse goal, such as indexes[1] considered, or importance of index. For example, for the retrieval of an adequate solution strategy for a problem, relevant indexes might be the problem description and the task when the problem occurred, whereas potential problems during a specific task might be identified based on the task only.

Dealing with Incomplete Data. It may be impossible to describe the current situation completely, in order to search for relevant SE experienceware. For example, when beginning a software project, information such as size of software system, may not yet be available. Also stored experiences might be incomplete, e.g., some features were unknown in the past, or new features have become relevant for the retrieval due to changes of the software environment. For example, when a company starts to use different inspection techniques, the feature «type of inspection» may become important in order to discriminate quality models on fault detection rates. This potential incompleteness of data has to be explicitly considered when determining the similarity of experiences.

Relevance of Indexes for Similarity. The retrieval of experienceware from the EB is done based on a set of indexes, such as, e.g., application domain or platform. The problem here is to identify a set of indexes that predicts the usefulness of the experience in a specific situation. This is further complicated through the common lack of domain models in the software engineering domain and differences between software organizations, e.g., in one organization the type of inspection technique may influence the number of faults detected, whereas it does have no influence in another one. The relevance of a certain index also depends on the specific retrieval goal. For example, in order to identify relevant quality models, a correspondence with the measurement goal might be more important than with the system platform. Thus, initially we can only describe hypotheses about correlations between SE experiences, contexts, and retrieval goals, which have to be continuously tailored to the specific environment.

Local Similarity Measures. Similarity can be judged on two levels: global similarity measures are based on the number of facts that the past and the new situation have in common. In the software engineering domain, this may often lead to the rejection of cases as reuse candidates, because only a few of their index values are identical to the given situation, although two cases might be quite similar. For example, searching for experiences in relation to C used as a programming language, also experiences related to the use of C++ might be of interest. Thus, the determination of similarity has to be extended to local similarity between values of the indexes.

These requirements demonstrate that sophisticated ways are required for the retrieval of useful and appropriate SE experienceware.

3. Determination of Retrieval Goals

Based on reuse scenarios, retrieval goals are determined. Retrieval goals explicitly state the object to be reused, the purpose of reuse, the related task, the specific viewpoint, and the particular environment (see Table 18).

Goal Template	Example
Retrieve <object>	lesson learned
for the <purpose>	guide problem solving
concerning <process>	software measurement

Table 18. Retrieval goal template

1. As index we denote attributes of the case that predict the usefulness of the case concerning the given situation description and that are used for retrieval and determination of the similarity value.

from the <viewpoint>	quality assurance personnel
in the context of <environment>	company IntelliCar

Table 18. Retrieval goal template

Based on the retrieval goals, reusability factors are determined through interviews with domain experts. This includes the specification of related concepts, relevant indexes, and their interdependencies. Objects defined in the retrieval goal, are explicitly described in a concept glossary [28] stating their intended purpose and users. Relevant attributes of the objects used as indexes in the retrieval process are identified, based on the following considerations wrt. the specific retrieval goal:

- Which are characteristic attributes to describe the object?
- Which information are required for the execution of the task concerning the specific purpose?
- Which information are required for the identification and selection of adequate objects?
- Which attributes enable to discriminate objects?
- Which minimal quality the objects must have?

For each of the identified indexes, its type, range and cardinality are explicitly defined [28] in order to allow the consistent knowledge representation and to support the situation assessment.

For the retrieval of SE experienceware a generic similarity measure is developed, which is tailorable to each retrieval goal, considering global and local similarity, relevance of indexes, and incompleteness of knowledge (see Section 5).

4. Retrieval of Software Engineering Experienceware

In order to provide experience-based support to SE tasks, the retrieval process includes the following steps:

Step 9. Situation assessment. The current situation is described by the user specifying the retrieval goal and a set of indexes related to the retrieval goal. The importance of each index wrt. the specific reuse goal is stated through a predefined relevance factor assigned to each index, which can be manually adapted to the specific situation by the user (see Section 5.3). Figure 2 illustrates a situation assessment with an exemplary set of indexes.

Step 10. Exact matching of indexes marked as essential. In a first step, the cases of the EB are matched with the given situation description wrt. the indexes marked as essential. By perfectly matching those indexes, a set of potential reuse candidates is determined. Figure 2 shows a simplified example: while comparing the cases of the EB with the situation assessment, Case_03 and Case_11 are considered as potential reuse candidates, because the values of the indexes marked as essential are equal to the present ones. Case_07, which describes an experience regarding the development of the measurement plan, is not further considered, as the value of the index «task» is different to the one of interest.

Step 11. Partial matching of cases. For all potential reuse candidates, a similarity value is computed by partially matching the indexes (except the ones marked as essential) using a specific similarity measure wrt. the retrieval goal (see Section 5). Cases with a higher similarity value than a given threshold are considered as sufficiently similar and the most similar cases are proposed to the user as reuse candidates. Continuing the example shown in Figure 2, Case_03 is considered more similar than Case_11 to the given situation, as the values of the indexes of Case_03 marked as important or less important are more similar to the current ones.

Step 12. Selection of reuse candidate(s). Based on the proposed reuse candidates associated with their similarity value, the user can explore the cases via browsing and navigation along their relationships, select the most appropriate case(s) for reuse and, if necessary, adapt them to fit the current needs. Selection of most useful reuse candidates. Additional information included in cases concerning their reuse in the past, e.g., dates of reuse, adaptations done, or reuse cost further guide the selection and adaptation process.

To identify useful experienceware, we need a flexible similarity measure (see step 3), considering the specific requirements as described in Section 2.

5. Similarity Measure for Retrieving Software Engineering Experienceware

In this section, we define a generic similarity measure (i.e., a parameterized set of similarity measures) for the identification of similar SE experienceware in the EB that can be tailored to a specific reuse goal. The similarity measure is used to assign a similarity value (defined as a real number $\in [0,1]$) to the stored cases by matching them partially to a given situation description (see Section 4.). Basically, a similarity measure sim(x,y) can be defined by [4,25,30]:

A mapping sim: M x M-> [0,1] on a set M is a similarity measure iff $(\forall x,y \in M)$:

- sim(x,x)=1 (reflexivity)
- sim(x,y)=sim(y,x) (symmetry)[1]
- sim(x,y) = 1 <-> x=y

Depending on the retrieval goal, a particular set of indexes is used for assessing the new situation and matching it to the cases stored in the EB. A set of indexes C is represented as a list of features $C=\{C_1, C_2,...\}$. The range of the value c_i of the feature C_i is defined by W_i, its respective range definition stored in the EB (see Figure 3). The present situation is assessed based on the set of indexes wrt. the retrieval goal. The description of the present situation is represented as a list of feature-value pairs $Sit=\{(C_1, s_1), (C_2, s_2),...\}$ including the features $C_i \in C$ and their values $s_i \in W_i$ (see Figure 3). In the EB, a $case_k = (E_k,e_k)$ represents an experience by feature-value pairs (experience $E_k=\{(E_{k1}, e_{k1}), (E_{k2}, e_{k2}),...\}$ with the features E_{ki} and their values $e_{ki} \in W_i$, describing the know-how gathered in a past software project, the context from

1. For our purposes, we assume a similarity measure to be symmetrical, as it is used for the computation of a similarity value of a case of the EB in reference to a given situation assessment. However, in general, similarity measures do not need to be symmetrical.

Situation Assessment

Reuse goal		
object	lesson learned	
purpose	guide solving of problem	
process	sw measurement	
viewpoint	quality assurance personnel	
environment	*IntelliCar*	
Indexes		
department	irrelevant	ABS
staff size	less important	10
application domain	essential	automobile
improvement goal	important	improvement of sw system reliability
programming language	irrelevant	Ada
dev. experience	less important	high
sw system size	less important	unknown
measurement maturity	important	initial
task	essential	measurement goal definition

Measurement -EB (excerpt)

CASE	Case_003	FS	Case_007	FS	Case_011	FS
organization	FuelInjection	–	FuelInjection	–	FuelInjection	–
staff size	15	E	100	W	50	W
application domain	automobile	–	automobile	–	automobile	–
improvement goal	sw system reliability improvement	E	sw system reliability improvement	E	development cost reduction	W
prog. language	Fortran	–	Ada	–	C	–
dev. experience	medium	E	low	W	low	W
sw system size	15 KLOC	U	80KLOC	U	60KLOC	U
msmt maturity	_[a]	R	–	R	–	R
task	measurement goal definition	–	measurement plan development	–	measurement goal definition	–

a. Comment: Attribute was not captured in the past.

Figure 2. Simplified retrieval example

which its originates, and its interdependencies with other products, processes, and resources and a threshold $e_k \in [0,1]$ that defines a threshold of the global similarity value determining if the case is sufficiently similar to the given situation to be proposed to the user as a reuse candidate.

5.1 Goal-Oriented Retrieval

In most CBR systems, indexes are defined as a fixed part of the case stored in the case base. As in the software engineering domain, indexes vary depending on the particular retrieval goal, indexes are related to the specific type of retrieval. A specific retrieval goal g is associated with a particular set of indexes $Cg=\{Cg_1, Cg_2,...\}$. Therefore, a generic similarity measure is defined using a set of indexes as a parameter that can be instantiated wrt. the specific retrieval goal.

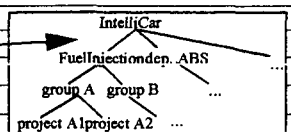
5.2 Dealing with Incomplete Data

As today software engineering domain models are usually not available in practice, we often have to deal with incomplete information, concerning the input as well as the cases stored in the case base. Our approach is based on Tversky's contrast model [29] as applied in PATDEX [7]. This model expresses similarity of two objects through a linear contrast of weighted differences between their common and different features. Concerning the potential situations that can occur when comparing the given situation and a stored case the following feature sets are distinguished:

- E: Set of corresponding features. The value of the feature of the given situation corresponds with the one of the stored case. For example, if both, the situation assessment and the stored case state the feature «experience of developer» as high.
- W: Set of contradicting features. The value of the feature of the given situation does not correspond with the one of the stored case. For example, if in the past no effort reporting tools were available, but now in the given situation the feature «effort reporting tools» is stated as available.
- U: Set of unknown features. The feature is only contained in the stored case, but not stated in the actual situation description. For example, when initiating a software project certain information, such as «software system size» may be stated as unknown in the situation description, although available in the stored cases on past projects.
- R: Set of redundant features. The feature is stated in the given situation description, but not contained in the stored case. Cases may have been stated incompletely in the past for various reasons, e.g., time pressure or lack of domain knowledge. For example, during the continuous organizational learning process, the feature «developer experience» may not have been considered initially, but later become important for the identification of relevant cases.

Figure 2 shows an example on the feature sets (FS) of the cases of the EB for a given situation description. This subdivision into different feature sets allows the explicit handling of unknown information and corresponding/contradicting features during the retrieval process by associating with each of these feature sets a different weight ($\alpha, \beta, \gamma, \delta \in [0,1]$). Depending on if we choose an optimistic or pessimistic strategy, we can weigh the corresponding features stronger than the contradicting ones or vice versa, respectively. In the software engineering domain, where companies are today beginning to build up EB's and a rather small amount of experience available, we suggest an optimistic strategy with $\alpha > \beta$, weighting corresponding features stronger than non-corresponding ones. Unknown features are considered as less important for the identification of relevant cases ($\gamma=0$). Redundant features are believed to have an impact on the determination of relevant cases. But as the specific values are not available in the respective case in the case base, they are associated with a very small weight ($\delta << \beta$). The weights depend on the specific en-

Example Index Set		Type and Range Definition
C_1	organization	Taxonomy
C_2	org. msmnt infrastructure	Binary symbol
C_3	staff size	Interval of number [0,100]
C_4	improvement goal	String
C_5	application domain	Unordered symbol: {automobile, airplane}
C_6	programming language	Unordered symbol: {Ada, C, C++, Fortran}
C_7	developer experience	Ordered symbol: {none, medium, high, expert}
C_8	sw system size (KLOC)	Number: [0,100]
C_9	project duration (month)	Number: [0,100]
C_{10}	measurement maturity	Ordered symbol: {initial, low, routine}
C_{11}	task	Unordered symbol: {msmnt goal def, msmnt plan dev, data collection}



Present Situation	
s_1	ABS
s_2	yes
s_3	[8,10][a]
s_4	"improvement of system reliability"
s_5	automobile
s_6	Ada, C
s_7	high
s_8	unknown
s_9	12
s_{10}	low
s_{11}	msmnt goal definition

a. The project team size varies from 8 to 10 developers during the software project.

Figure 3. Example of index set

vironment and may change during the life cycle of the EB, e.g., when the number of available experienceware cases increases a more pessimistic strategy might be more beneficial to avoid the retrieval of a too extensive amount of cases.

5.3 Relevance of Indexes for Similarity

The retrieval of experienceware from the EB is only useful, if the retrieved experienceware can actually be applied in the given situation. The problem here is to identify relevant features for the retrieval of appropriate cases, before trying to reuse this experience in the present situation. Based on PATDEX [7], relevance factors are defined, which determine the importance of a feature. However, in the SE context relevance factors have to be related to the specific retrieval goal. Thus, for each index $Cg_i \in$ index set Cg for a specific retrieval goal g, a relevance factor $\omega g_i \in [0,1]$ is defined. For each retrieval goal, those relevance factors are represented by means of a relevance vector $Rg = \{\omega g_1, \omega g_2, ...\}$ with $\sum \omega g_i = 1$ normalized. The relevance factor represents a weight used for the calculation of the similarity value. The relevance factors can be learned over time by a pseudo connectionistic approach [4] based on an abstracted Hebb competition learning taken from pattern recognition. Setting initially all indexes equally relevant, during the learning process the weights of features receiving a positive feedback are strengthened or weakened in case of negative feedback.

In addition, the user can manually overwrite the stored relevance factors, if necessary. This is facilitated through the classification of the factors into the following categories and associated weights:

- essential. Index has to exactly match the one stated. Through the declaration of essential indexes the user defines filter or knock-out features. Cases whose values of the features defined as essential do not exactly match are rejected regardless of the values of the other features.
- important/less important. Index is partially matched with the situation description. For example, setting $\omega g_i = 0.7$ if $C_i :=$ important, $\omega g_i = 0.3$ if $C_i :=$ less important.
- irrelevant. Index is disregarded for retrieval ($\omega g_i = 0.0$).

Systematically capturing feedback on manual modifications of relevance factors through the user enables the continuous learning and tailoring of the relevance vectors, e.g., by disregarding features frequently marked as irrelevant of the index set.

5.4 Local Similarity of Features

For the comprehensive examination of the similarity between a given situation and the cases in the EB, local similarity between the values of a feature has to be considered. Thus, specific local similarity measures have to be defined for each index type, e.g., numerical, symbolic, or textual in dependence on the specific organization. Here, generic local similarity measures $\upsilon'(v_i,v_j) \in [0,1]$ for basic value types W(v) in the software engineering domain [19,28] are provided, which still have to be refined for the specific environment in practice.

Let v_i and v_j be two values of a feature: v_i represents the value of the given situation and v_j the respective value of a stored case in the EB. Besides representing features by atomic values, e.g., the value «high» of the feature «developer experience», it may be necessary to represent features by sets of values. For example, the usage of different programming languages may be described through the set {C, Fortran, Assembler} of symbolic values. Thus, feature values are interpreted as sets with their minimal and maximal number of elements defined by their cardinality. The local similarity value is computed based on the normalized sum of best matches for each element of the value set in the situation assessment with the case [28]:

$$\upsilon'(vi,vj) = \sum_{e2 \in vj} (\max (\upsilon(e_1,e_2) \mid e_1 \in v_i)) / \text{card}(v_j)$$

if $\text{card}(v_i) > 0$ and $\text{card}(v_j) > 0$,

$\upsilon'(v_i,v_j) = 0$ otherwise.

In the following sections, local similarity measures $\upsilon(v_i,v_j)$ for the computation of a similarity value between two atomic feature values are described for basic types [17].

5.4.1 Numbers. Similarity between numeric values can be described as an asymptotic function [9], where differences between the two values cause a asymptotic decrease of similarity. To tolerate differences between the values until a defined distance x_1 the following similarity function can be

used: $\upsilon(v_i,v_j) = \exp(-\alpha \mid v_i - v_j\mid)^3$ with $\alpha \in \Re$.

5.4.2 Interval. The similarity of two features described by intervals can be calculated by regarding their borders [28]. Be $v_i=[l(v_i), h(v_i)]$ and $v_j=[l(v_j), h(v_j)]$ intervals:

$\upsilon(v_i,v_j) = (l(v_j)-l(v_i)+l(v_j)-h(v_i))/2$ if $h(v_i)<l(v_j)$,
$\upsilon(v_i,v_j) = (l(v_j)-l(v_i))/2$ if $l(v_i)<l(v_j)\leq h(v_i)\leq h(v_j)$,
$\upsilon(v_i,v_j) = 1$ if $l(v_j)<l(v_i)\wedge h(v_i)\leq h(v_j)$,
$\upsilon(v_i,v_j) = (h(v_i)-h(v_j))/2$ if $l(v_j)<l(v_i)\leq h(v_j)\leq h(v_i)$,
$\upsilon(v_i,v_j) = (l(v_i)-h(v_j)+h(v_i)-h(v_j))/2$ if $h(v_j)<l(v_i)$,
$\upsilon(v_i,v_j) = (l(v_j)-l(v_i)+l(v_i)-h(v_j))/2$ otherwise.

5.4.3 Binary Symbol. The similarity measure for binary symbols is defined by:

$\upsilon(v_i,v_j) = 1$ if $v_i=v_j$, and $\upsilon(v_i,v_j) = 0$ otherwise.

5.4.4 Ordered Symbol. The distance between the values of the list can be expressed by associating ranking numbers with each value, assuming an equidistant order of the values, or through user-defined numerical values. Based on the associated numerical values, the similarity functions as described for numbers can be used for the determination of the similarity value between ordered symbolic values.

5.4.5 Unordered Symbol. For the determination of similarity between values of a finite unordered list of symbols, the similarity measure can be defined in form of a triangular similarity matrix S_{ij} that assigns every combination of values $v_i,v_j \in W(v)$ a respective similarity value $\upsilon(v_i,v_j) \in [0,1]$ as specified in a specific environment.

5.4.6 Taxonomy. A similarity measure for taxonomies is based on the position of objects by assigning similarity values $SV_i \in [0,1]$ to each inner node K_i of the taxonomy, such that the following condition holds: if K_j is a succcssor of K_i then $SV_i < SV_j$. Be K_i, K_j nodes of the taxonomy, $<K_i, K_j>$ the node that is the nearest predecessor of K_i and K_j, and $SV<K_i, K_j>$ the similarity value assigned to the node $<K_i, K_j>$. The similarity measure of the nodes K_i, K_j is then defined by $\upsilon(K_i, K_j) = 1$ if $K_i = K_j$, or $\upsilon(K_i, K_j) = SV<K_i, K_j>$ otherwise [11].

5.4.7 String. An approach to determine the similarity can be based on the greatest common substring-string ratio [9]:

$\upsilon(v_i,v_j) = 0$ if $|v_i|=0$,
$\upsilon(v_i,v_j) = \Sigma|cv_j| / \Sigma|w_i|$ otherwise, with $\max|cv_j|$ length of common substring per word w_j of $v_i \wedge (|cv_j|>3) \wedge (|w_i|>3)$[1].

5.5 Similarity Threshold

When an EB only contains few cases, as probably in the beginning of building an EB, many cases with a very low similarity value may be retrieved due to the lack of more appropriate alternatives. To prevent overloading the user with less useful information, cases with a small similarity value are excluded by setting a threshold. The threshold explicitly states when cases are to be considered to be sufficiently similar. Global thresholds $e_k \in [0,1]$ as part of the case description $case_k = (E_k,e_k)$ define the lower border the global similarity value has to exceed to consider this specific case as a reuse candidate. The local similarity threshold $\theta_i \in [0,1]$ defines for index Cg_i if two values are considered as (sufficiently) similar.

The thresholds can be updated based on the user feedback on the suggested reuse candidates. For example, if a case suggested as reuse candidate has been rejected several times by the user the corresponding threshold can be increased, impeding the suggestion of this case as reuse candidate only with higher similarity values in the future.

5.6 Similarity Measure for the Retrieval of Software Engineering Experienceware

Considering the specific requirements given in the software engineering domain as described in the previous sections, we define a similarity measure sim(Sit',Ek'). The similarity measure is based on the following assumptions:

- Given a specific retrieval goal g,
- Set of indexes $Cg=\{Cg_1, Cg_2,...\}$ wrt. the retrieval goal g,
- $Sit'=\{(Cg_i, s_i) \in Sit \mid \text{relevance factor } (S_i) \neq \text{essential}\}$,
- $case_k = (E_k,e_k)$ of the case base, with $E_k' \subseteq E_k \wedge \forall\ E_{ki}' \in Cg \wedge$ features $E_{ki}' \in Cg$ and their respective values e_{ki}',
- local similarity value $\upsilon'(s_i,e_{ki}')$ of the features s_i and e_{ki}',
- weight ωg_i of the relevance vector Rg wrt. retrieval goal g,
- local threshold $\theta_i \in [0,1]$ for index Cg_i,
- The feature sets with the weights $\alpha, \beta, \gamma, \delta \in [0,1]$:
  $E = \{Cg_i \mid (Cg_i \in Sit' \cap E_{ki}')$ and $(\upsilon'(s_i,e_{ki}') \geq \theta_i)\}$
  $W = \{Cg_i \mid (Cg_i \in Sit' \cap E_{ki}')$ and $(\upsilon'(s_i,e_{ki}') < \theta_i)\}$
  $U = \{Cg_i \mid (Cg_i \in E_{ki}' - Sit')\}$
  $R = \{Cg_i \mid (Cg_i \in Sit' - E_{ki}')\}$

⇨The global similarity measure is defined as:

$$sim(Sit',Ek') = (\alpha \Sigma_{si \in E}\, \omega_{ik} \upsilon'(s_i,e_{ki}')) / ((\alpha \Sigma_{si \in E}\, \omega_{ik} \upsilon'(s_i,e_{ki}')) + (\beta \Sigma_{si \in W}\, \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))) + (\gamma \Sigma_{si \in U}\, \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))) + (\delta \Sigma_{si \in R}\, \omega_{ik}(1-\upsilon'(s_i,e_{ki}'))))$$

$Case_k$ is considered as reuse candidate, if all features marked as essential in the given situation exactly match the respective features of the case and $sim(Sit',E_k') \geq$ global similarity threshold e_k of $case_k$.

6. Continuous Evolution of Retrieval

Based on feedback from the application of the retrieval in a specific environment, the retrieval method can continuously be adapted to the specific characteristics of the environment, improving the retrieval performance. Protocols documenting the users (re-)actions and user-provided critics and suggestions can serve as a basis for the maintenance through the knowledge engineer. For example, consideration of new attributes for retrieval through the user can indicate the need of the integration of those attributes in the indexing scheme. Manual modifications of the relevance factors can imply the modification of weights (see Section 5.3) or even the removal of indexes, if they are frequently considered as irrelevant by the user. Variations concerning the number of retrieved reuse

1. Here only substrings > 3 characters are considered in order to filter articles and prepositions.

candidates, can imply a different retrieval strategy, changing, e.g., the weights related to the feature sets (see Section 5.2) or the tresholds (see Section 5.5). Observing the frequent rejection of suggested cases, can point at required modifications, e.g., if only a specific case is concerned, we can increase the global threshold for the case (see Section 5.5). Based on the further critics and suggestions from the user, the indexing scheme and similarity measure can be carefully reviewed.

7. Discussion

In literature various approaches exist for experience-based support for the planning and execution of software projects, primarily focusing on software code, e.g, based on library and information science, knowledge-based systems, or database management technologies [15]. However, the majority of those approaches fails to recognize the complexity of SE experienceware in general, often requires a thorough classification of the domain, or does not provide any means for similarity-based retrieval. Recently, CBR has been recognized as a promising approach for the operationalization of learning organizations in the SE domain [2,3,19,23,26]. Applications are developed in different areas, like capturing est practices (e.g., [5,21]), effort prediction (e.g., [16]), measurement [18,20], or change management [24]. The retrieval effectiveness of those systems varies from simple browsing facilities to similarity-based retrieval methods. However, the majority of those systems does not explicitly consider different reuse purposes, nor offers flexible retrieval methods and similarity measures tailorable to the specific goals.

In this context, an advanced CBR approach is PATDEX [7] that has been developed for the fault diagnosis of CNC machining centers, an analogous situation to the reuse of SE experienceware. PATDEX uses a flexible similarity measure explicitly dealing with unknown information, filter attributes, and local similarities, which forms the basis of a similarity model comparable to our approach. In PATDEX, the case-based test selection (focusing on "find the most relevant attribute for value determination") and case-based diagnosis (focusing on "find the most similar diagnosis") have been separated, by creating distinct case bases, one containing strategy cases for the selection of tests and a second one containing diagnosis cases. This can be seen in analogy to different retrieval goals. However, due to administration and maintenance reasons, it would be inefficient to build different, partly redundant case bases for different retrieval goals. Thus, as a single case (or parts) can be useful for several purposes, the retrieval method and the similarity measure have to be flexible and tailorable to the specific goal.

Some approaches involve an explicit concept of context in the retrieval process, for example, INRECA includes a context concept as a "similarity environment for the retrieval" [1,30]. Every context enables a specific view on the particular application domain and relates the view with specific properties. Enhancing the view concept from database technologies, a context does not only encapsulate static information for the definition of objects, but also information on their dynamic processing. Contexts can be defined and selected statically (during the knowledge acquisition phase) or dynamically (by the user during runtime). However, so far, the concept has not been defined and implemented completely in the system.

Until now, only a few approaches allow a flexibilization of similarity determination for specific applications or classes of applications, for example, through dynamic ranking of importance ratings of indexes [22] in dependence of a specific retrieval context. ReMind enables the partitioning of the case base through the use of prototypes [8] and the assignment of a different set of importance values to each partition. However, if multiple retrieval goals have to be supported by a case base, then partitioning is not sufficient, because the cases potentially contribute to different retrieval goals (see argumentation above wrt. the maintenance and administration). Another approach is the association of one set of relevance criteria to each case, when cases are highly idiosyncratic. The set of importance values associated to a case emphasizes the combination of those dimensions of the cases that have been utilized in the past to judge the usefulness of a case. However, as in our application, multiple retrieval goals have to be supported by a case base, the evaluation of cases depends on the specific retrieval goal, i.e., in general the evaluation of a case differs for different retrieval goals. PATDEX models relevances of attributes wrt. to a certain diagnosis, represented in a symptom-diagnosis matrix. As cases in PATDEX are classified diagnosis examples (each case represents a particular diagnosis), a specific relevance vector is assigned to each case, modeling the influence on the retrieval of the respective correct diagnosis. This enables the decomposition of the general retrieval goal "find diagnosis" into the subgoals "find/confirm diagnosis 1..n". However, SE experienceware cases represent unclassified examples. Only the introduction of retrieval goals, as in our approach, which models the importance of particular attributes concerning a specific retrieval goal, allows the transfer of those techniques from PATDEX.

CBR-Works [27], one of the commercial successors of INRECA, offers a variety of possibilities to filter attributes of a query or to change weights dynamically. However, it does not explicitly model the concept of different retrieval goals.

In contrast, our approach systematizes the concept of goal-oriented retrieval through a flexible and tailorable retrieval method and similarity measure wrt. the specific goal. Different sets of indexes, their importance, and the similarity measure are associated wrt. the specific goal, indicating how useful experienceware can be retrieved in the particular situation.

Many CBR systems evaluate similarity between cases only on the global case level. Based on the similarity model used in PATDEX, our approach also considers local similarities between single attribute values. Basic local similarity measures for different types of indexes in the software engineering domain have been developed, and the concept of a local threshold, defining similarity borders on the attribute level has been

integrated.

Another issue of essential importance in the software engineering domain is the continuous evolution of the retrieval process and the similarity measure based on feedback from its application in practice. Only a few systems offer learning possibilities regarding the similarity measure as, e.g., the tailoring of relevance factors (see [4] for an overview), which represent the basis for the learning possibilities of our approach.

8. Conclusions

For the successful reuse of software engineering know-how in practice, useful and appropriate experienceware has to be retrieved from a corporate memory. In this paper, we propose a flexible goal-oriented retrieval method and tailorable similarity measure for the effective and efficient retrieval of SE experienceware based on our experiences on reuse of software measurement experienceware in industrial projects. The concept of goal-oriented retrieval is systematized, which enables the retrieval of experienceware cases of one EB for different goals. The approach shows how CBR systems, which can handle different retrieval goals, become more flexible and can deal with a broader class of problems.

Our approach is incorporated in the REMEX system, a prototypical EF for the experience-based support of the planning of software measurement programs [20] currently implemented. Further empirical research has to be carried out to assess strengths and weaknesses of the approach. Based on feedback from its application in practice we expect a basis for future research on methods for the guidance of tailoring the similarity model in industrial environments.

9. References

[1] K.-D. Althoff et al. Case-Based Reasoning for Decision Support and Diagnostic Problem Solving: The INRECA Approach. Proc. 3rd German Workshop on Case-Based Reasoning, 1995.

[2] K.-D. Althoff et al. CBR for Experimental Software Engineering. In M. Lenz et al. (eds.), Case-Based Reasoning Technology - From Foundations to Applications, Springer Verlag, 1998.

[3] K.-D. Althoff, F. Bomarius, C. Tautz. Using Case-Based Reasoning Technology to Build Learning Software Organizations. Proc. of Workshop on Building, Maintaining, and Using Organizational Memories at the 13th European Conference on AI, 1998.

[4] K.-D. Althoff. Evaluating Case-Based Reasoning Systems: The Inreca Case Study. Postdoctoral Thesis (Habilitationsschrift), University of Kaiserslautern, Germany, 1997.

[5] K.-D. Althoff, M. Nick, C. Tautz. CBR-PEB: An Application Implementing Reuse Concepts of the Experience Factory for the Transfer of CBR System Know-How. Proc. of the 7th German Workshop on Case-Based Reasoning, Germany, to appear 1999.

[6] A. Aamodt, E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 17(1), 1994.

[7] K.-D. Althoff, S. Wess. Case-based Knowledge Acquisition, Learning and Problem Solving in Diagnostic Real World Tasks. Proc. of the 5th European Knowledge Acquisition for Knowledge-Based Systems Workshop, Scotland/UK, 1991.

[8] R. Barletta. A Hybrid Indexing and Retrieval Strategy for Advisory CBR Systems Built with ReMind. Proc. of the 2nd European Workshop on Case-Based Reasoning, 1994.

[9] R. Bergmann et al. Initial Methodology for Building and Maintaining a CBR Application. ESPRIT Project 22196, 1997.

[10] V. R. Basili, G. Caldiera, H. D. Rombach. Experience Factory. In John J. Marciniak (ed.), Encyclopedia of Software Engineering, vol. 1, John Wiley & Sons, 1994.

[11] R. Bergmann. On the Use of Taxonomies for Representing Case Features and Local Similarity Measures. Proc. of the 6th German Workshop on Case-Based Reasoning, Germany, 1998.

[12] J.M. Barr, R.V. Magaldi. Corporate Knowledge Management for the Millennium. In I. Smith, B. Faltings (eds.), Advances in Case-Based Reasoning, Springer Verlag, 1996.

[13] A. Birk, C. Tautz. Knowledge Management of Software Engineering Lessons Learned. Proc. of 10th Int. Conference of Software Engineering and Knowledge Engineering, San Francisco, 1998.

[14] CEMP Consortium. Customized Establishment of Measurement Programs. Final Report, ESSI Project Nr. 10358, 1996.

[15] W. B. Frakes, P. B. Gandel. Representing Reusable Software. Information and Software Technology, 32(10), 1990.

[16] G. R. Finnie, G. W. Wittig, J.-M. Desharnais. Estimating Software Development Effort with Case-Based Reasoning. Proc. of the 2nd Int. Conference on Case-Based Reasoning, RI, July 1997.

[17] C. Gresse von Wangenheim, K.-D. Althoff, R. M. Barcia. Intelligent Retrieval of Software Engineering Experienceware. Technical Report PPGEP-C3001.99E, Graduate Program in Production Engineering, Federal University of Santa Catarina, Brazil, 1999.

[18] C. Gresse von Wangenheim et al. Case-Based Reasoning Approach to Reuse of Experiential Knowledge in Software Measurement Programs. Proc. of the 6th German Workshop on Case-Based Reasoning, Germany, 1998.

[19] C. Gresse von Wangenheim. Knowledge Management in Experimental Software Engineering - Create, Renew, Build and Organize Knowledge Assets. Proc. of the 10th Int. Conference on Software Engineering and Knowledge Engineering, San Francisco, 1998.

[20] C. Gresse von Wangenheim, A. von Wangenheim, R. M. Barcia. Case-Based Reuse of Software Engineering Measurement Plans. Proc. of the 10th Int. Conference on Software Engineering and Knowledge Engineering, San Francisco, 1998.

[21] S. Henninger. Capturing and Formalizing Best Practices in a Software Development Organization. Proc. 9th Int. Conference on Software Engineering and Knowledge Engineering, Spain, 1997.

[22] J. L. Kolodner. Case-Based Reasoning. Morgan Kaufmann, San Francisco, California, 1993.

[23] H. Kitano, H. Shimazu. The Experience-Sharing Architecture. In D. Leake (ed.), Case-Based Reasoning Experiences: Lessons Learned & Future Directions, 1996.

[24] W. Lam, V. Shankararaman. Managing Change During Software Development: An Incremental, Knowledge-Based Approach. Proc. of the 10th Int. Conference on Software Engineering and Knowledge Engineering, San Francisco, 1998.

[25] M.M. Richter. On the Notion of Similarity in Case-Based Reasoning. G. della Riccia et.al (eds.), Mathematical and Statistical Methods in Artificial Intelligence, Springer Verlag, 1995.

[26] C. Tautz, K.-D. Althoff. Using Case-based Reasoning for Reusing Software Knowledge. Proc. of the 2nd Int. Conference on Case-Based Reasoning, LNAI 1266, Springer, 1997.

[27] CBR-Works. TecInno GmbH, Germany.

[28] C. Tautz, C. Gresse von Wangenheim. REFSENO: A Representation Formalism for Software Engineering Ontologies. Proc. 5th German Conference on Knowledge-Based Systems, 1999.

[29] A. Tversky. Features of Similarity. Psychological Review, 84, 1977.

[30] S. Wess. Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik. Ph.D. Thesis, University of Kaiserslautern, Germany, infix Verlag, 1995.